3D Video Capture of a Moving Object in a Wide Area Using Active Cameras(Dissertation_全文)

AUTHOR(S):

Yamaguchi, Tatsuhisa

3D Video Capture of a Moving Object in a Wide Area using Active Cameras

Tatsuhisa Yamaguchi

Abstract

In this thesis, we tackle the problem of 3D video capture using active cameras. 3D video is a media that records dynamic visual events in the real world as is. 3D video data is full 3D representation of objects; 3D shape, its surface color and texture are generated from multi-view video taken by multiple video cameras that surrounds a target object.

In order to generate 3D video from multi-view video, every point on the object must be captured from different directions with high enough spatial resolution using accurately calibrated cameras.

Existing 3D video capture systems typically use a fixed set of cameras focused on a restricted space to satisfy all these requirements. Since a field of view of a camera is limited, such systems cannot be applied when the objects move in a large area. If we choose wide lens cameras to view the large space, details of object will not be captured by the cameras, and the spatial resolution of generated 3D video will be lowered.

Use of active cameras can capture an object that moves in a large area by changing its viewing direction and zoom to track the object. But there are two problems when considering 3D video capture using active cameras. The first one is accurate calibration of active cameras. The second is real-time tracking of object satisfying requirements for 3D video generation.

As a solution to these problems, we propose the "cell-based" framework. The main idea of ours is to divide the space where the object moves into topological subspaces named "cells" and resolve the problems for each cell. Namely, it first divides the space into cells, adjusts camera control values for each cell, calibrate cameras for each cell, then conduct object tracking based on the cells. The cell-based framework itself is not a single specific algorithm. Several specific algorithms for specific cases can be designed reflecting the type of object motion.

As an instantiation of the cell-based framework, we first propose a 3D video capture algorithm when the path of the object movement is given in advance. The

space along the path is divided into cells and the camera control is performed by a pseudo-optimal assignment of camera mode — which cell the camera should watch — to each point of the path. We also propose another algorithm to capture an object that moves freely on a flat floor. The algorithm divides the space into regular hexagons, then control the active cameras based on 3-coloring of the cells. We prove that capturing the nearest three cells to the object can ensure continuous object capture with at least one third of the cameras. Finally, we show an example design of a figure skater capture system based on a cell-based algorithm to show the scalability of our framework in a more realistic situation. The camera layout problem specific to the cell-based algorithm is discussed.

Acknowledgment

I would like to express my sincere appreciation to my advisor, Professor Takashi Matsuyama. He invited me to this interesting research and provided insightful advice in these six years.

I would also like to thank other members of my thesis committee, Professor Michihiko Minoh and Professor Yuichi Nakamura for taking the time to review my thesis and giving helpful comments.

Next, I would like to thank Dr. Atsuto Maki. He supported this work in its early stage when he was at Kyoto University.

I am deeply indebted to Senior Lecturer Hiroaki Kawashima, Senior Lecturer Shohei Nobuhara, Assistant Professor Tony Tung, Associate Professor Xuefeng Liang and all the members of Matsuyama laboratory, especially Hiromasa Yoshimoto and Tomohiro Mukasa.

Finally, I have to thank my family for everything they have done for me.

Contents

Chapter 1

Introduction

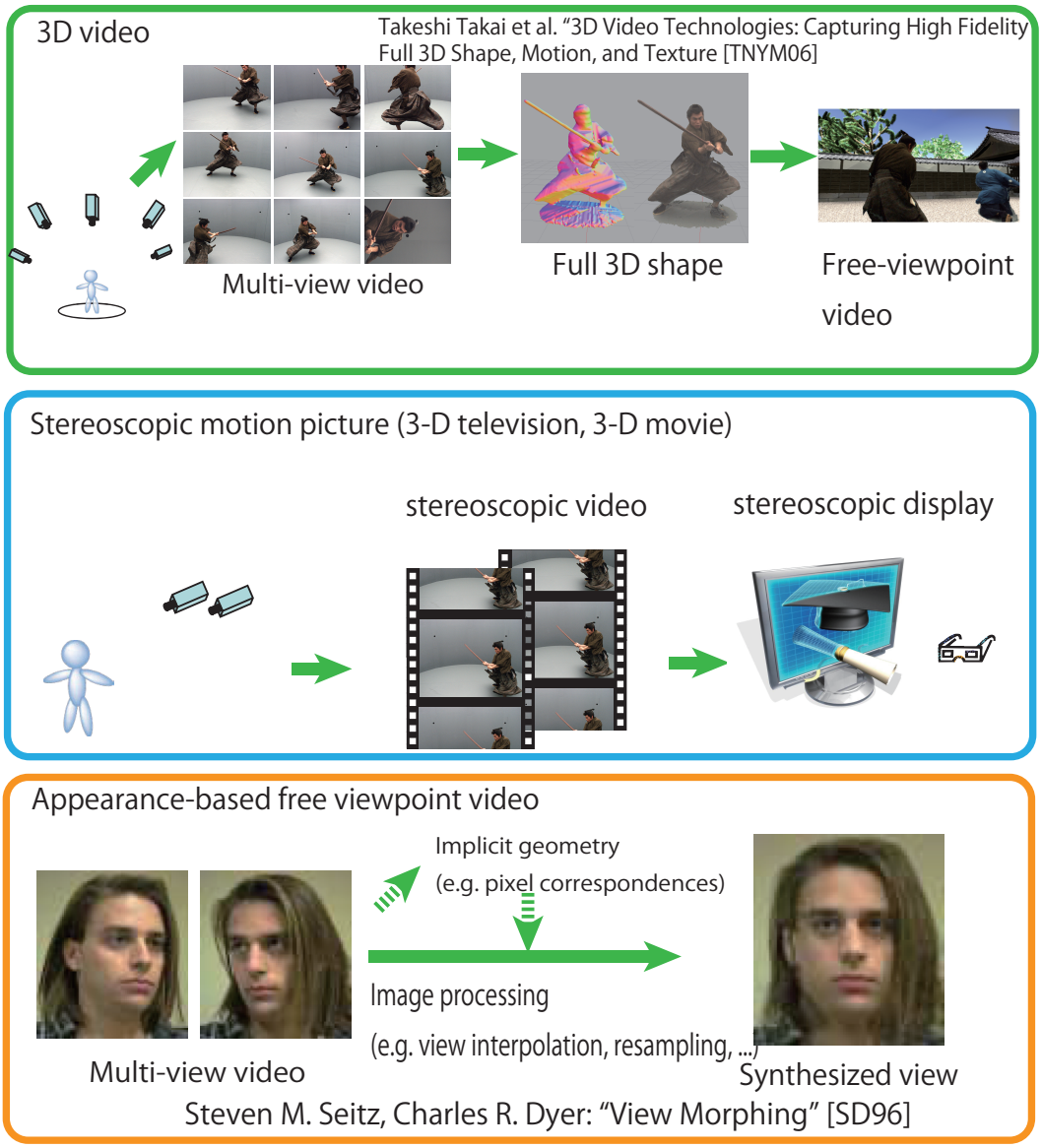
1.1 3D Video

Movie and video has been widely used to record visual events since the 20th century. In this early 21st century, 3D video was proposed as a new media that records dynamic visual events in the real world as is. Whereas ordinary 2D video just captures appearance of scenes, 3D video captures full 3D representation of objects, such as the 3D shape of an object and its surface color and texture.

3D video is often confused with stereoscopic motion pictures and free viewpoint video. The difference between 3D video and stereoscopic motion pictures is described in Figure 1.1. The term "3-D" [1] often refers to "stereoscopic" in the fields of home appliances, broadcasting or cinematography. For example, "3-D television" and "3-D movies" mostly refer to stereoscopic motion pictures or stereoscopic display devices. When two different video streams are presented to our right and left eyes, our brain regards the difference between two streams as the binocular parallax between the eyes, then perceives a 3D scene. However, stereoscopic motion picture data itself is just a pair of video streams. It does not have full 3D information of the scene. Namely, 3-D TV or 3-D movies are not full 3D media in this sense. We must distinguish such "3-D" media from 3D video.

Stereoscopic motion picture lacks the interactivity as well; we cannot change viewpoints of the scene. In the physical world, appearance of objects change as we change our viewpoints and viewing directions. If a media could resemble the appearance change when the viewer moves its viewpoint, it can give an illusion

[1] Both "3-D" and "3D" stands for "three-dimensional". Cinematographers tend to use "3-D" with a hyphen. On the other hand, we computer vision researchers basically use "3D" without a hyphen. The two notations are mixed in this section to emphasize the difference.



	3-D TV	Appearance-based	3D Video
Input images	Binocular	Dense	Sparse
Viewpoint	Fixed	Limited to nearby input views	Arbitrary
Scene edit	No	No	Yes
Relighting	No	No	Yes

Figure 1.1: Difference between 3-D television, appearance-based free viewpoint video and 3D video. Some pictures in "3D video" and "Stereoscopic motion picture" sections, designated "Multi-view video", "Full 3D shape", "Free-viewpoint video" and "stereoscopic video", are quoted from [TNYM06]. Three pictures in "Appearance-based free viewpoint video" section are quoted from [SD96].

of 3D perception by motion parallax. Such system is called free-viewpoint video (FVV).

There are several approaches to realize FVV. One of the approaches is called "geometry-based", which obtains full 3D representation of a scene by some measurement, then render the model from arbitrary viewpoints to synthesize new views.

Another approach is termed "appearance-based", which generates new virtual views from real views, based on image transformation such as morphing or interpolation, using quasi-3D geometry or no 3D geometry. For example, Light Field Rendering [LH96] composes a 4D function named light field from multi-view images, and then synthesizes 2D images from arbitrary viewpoints as the 2D slices of the light field. View Morphing [SD96] generates in-between views of two existing images by 2D transformation of images based on the pixel correspondences between the images. A drawback of appearance-based methods is the difficulty in generating realistic images when new viewpoint is largely different from those of input images. Moreover, some of the methods, such as View Morphing, limits the movement of the viewpoint on the line segment between two input viewpoints. Consequently, appearance-based methods requires more dense set of input images than geometry-based ones.

The geometry-based approach to FVV is almost equivalent to what 3D video does. However, 3D video is not just a way to realize FVV. Its full-3D data has further applications that is not feasible with 2D video. For example, analysis of a human shape model can give performer's motion description [WL01] like a vision-based motion capture. Compared to marker-based optical motion capture, 3D video has the advantage that it can capture human motion without the use of special markers or suits, and that it captures shape and texture of the performer as well. Another benefit out of 3D video is edit in its 3D representation. For example, when we add new objects into a 3D video stream, the lighting, shadows or occlusions can be automatically handled by the rendering process. Generating such physically-consistent images would be difficult in 2D video editing without much intervention by hand. In other words, 3D video facilitates scene editing, rather than video editing.

1.2 3D Video Capture using Multi-view Video

To my knowledge, no single camera can record 3D video by itself. Instead, a number of studies have proposed 3D video generation methods using multi-view video. One of the earliest works is found in Moezzi et al. [MTG97] Their method captures multi-view video of a performer in a 2m $\times$ 2m region using 17 cameras that surrounds the performer. Then a volumetric 3D model is generated from the images by a voxel-based volume intersection technique [Lau94]. The voxel model is converted into a triangular mesh model using Marching Cubes algorithm [LC87]. Finally, the colors on the vertices of the mesh model are computed based on the corresponding points on the original images.

After this work, several different 3D video capture methods have been proposed [ES04, FP06, KRN97, MWTN04, SH05]. These methods have followed the same methodology. Namely, most 3D video capture methods first capture objects by a set of multiple cameras, then reconstruct 3D shape of the objects based on the analysis of images using geometric and photometric properties of the cameras obtained separately.

Figure 1.2 describes the workflow of the 3D video capture and application. The process to generate 3D video can be separated into three parts.

1. Multi-view video capture
2. Camera calibration
3. 3D shape reconstruction

Additionally, the techniques listed below facilitate the use of 3D video media just like 2D video media.

4. Rendering
5. Data compression
6. Shape and motion analysis

1.2.1 Multi-view video capture

The first step of 3D video production is to capture target objects with a set of circumnavigating cameras. Several different types of multi-view video capture systems have been invented. A survey of the multi-view video capture studios

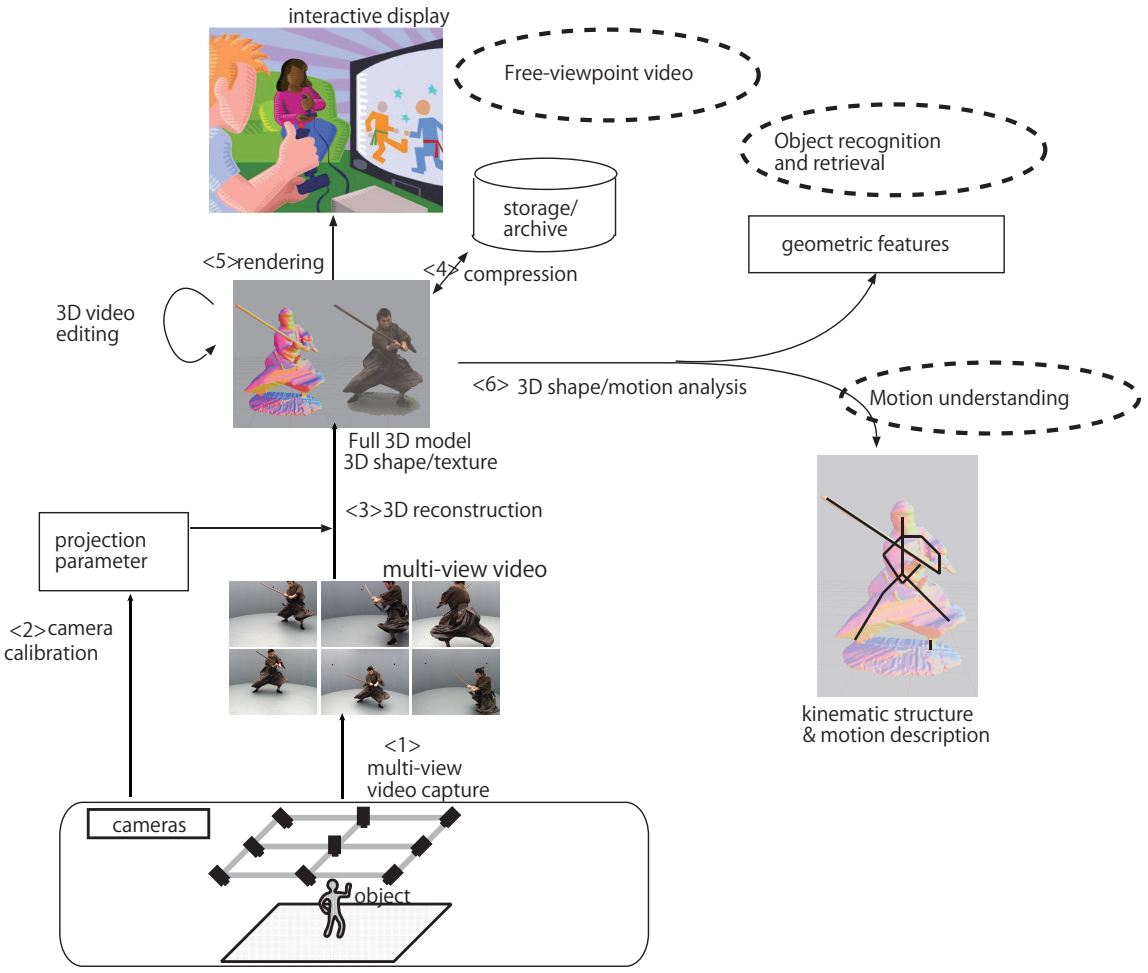


Figure 1.2: Detailed workflow of 3D Video and its applications. The pictures designated "multi-view video", "Full 3D model" and "kinematic structure & motion description" are quoted from [TNYM06] with some modification.

for 3D video production is shown in [SMN+09]. Most existing 3D video capture systems use static cameras to facilitate subsequent processes. Therefore a capture system should be designed depending on the object to be captured. In general, most of the systems have been designed for 3D video capture of human performer in a small space. For example, Moezzi et al. built a system for a 2m $\times$ 2m stage and achieved capture of a karate performer. Kanade et al. built a 5m geodesic dome with 51 cameras [KRN97]. An example of their result is a person swinging a baseball bat. In most cases, multiple cameras are placed at an equal distance apart from the center of the studio and are equally distributed in all the directions, such that a performer can be viewed from different directions regardless of its pose or direction.

1.2.2 Camera Calibration

Most 3D reconstruction algorithms require camera calibration as well as multi-view images. Camera calibration is a process to fit parameters of certain camera projection model to actual cameras. The camera calibration process in 3D video capture should obtain photometric and geometric parameters.

The photometric calibration associates amount of light energy entering to cameras with pixel values of camera images. It can be divided into intra-camera factors such as the vignetting, and inter-camera factors such as color response difference between cameras. For example, vignetting can be estimated by capturing a scene completely spanned by a uniform object surface[AAB96]. Color response curves of cameras can be estimated using reference objects such as the Macbeth Color Checker [IW05].

Geometric calibration concerns intrinsic and extrinsic parameters. Intrinsic parameters characterize the projection process determined by a camera and its lens. One of the widely used model is a modified pinhole camera model that can deal with the radial lens distortion [Zha00]. In this model, the intrinsic parameters consists of focal length, optical center of the image, scale and radial lens distortion coefficients. Zhang have also proposed a flexible method to obtain intrinsic camera parameter. A planar pattern, such as a chessboard pattern, is used as a reference object. The method takes a few images of the reference object under different orientations, then computes the camera parameters by the combination of a closed-form solution and non-linear optimization.

Extrinsic parameters represent the positions and orientations of multiple cam-

eras in a common world coordinate system. Extrinsic parameters of a single camera can be computed from correspondences between the coordinates of 3D points and their projection point in the image. In case of multiple cameras, 2D-to-2D correspondences between different camera images give clue to compute the relative positions and postures of those cameras. For example, eight-point algorithm [HZ04] is one of such algorithms. In a practical multi-view camera system, finding corresponding points in multiple images is non-trivial problem. In most cases, some artificial objects are used as calibration target so that the corresponding points between images are detected easily. For example, Svoboda et al. used a point light source to obtain 2D-to-2D correspondences robustly [SMP05]. A point light source can be observed from all directions and it is also easy to detect automatically.

1.2.3 3D Reconstruction

3D reconstruction in this thesis means to compute a geometric and photometric model of a scene, when given a set of images taken from multiple viewpoints with calibrated cameras.

Geometric Reconstruction

There are many methods to reconstruct 3D scene from a set of images. Those reconstruction methods can be classified by the type of information to use from images – silhouette, shading, textures, or features such as edges or vertices. Among these clues, most existing 3D video capture systems use silhouette first, then uses other clues to refine the result. One of the reasons why shape-from-silhouette is preferred is silhouette can be obtained relatively easily and robustly, and the shape-from-silhouette can give object shape robustly as long as silhouettes are available, regardless of object textures. Especially, most 3D video capture systems use artificial background e.g. uniform green screen for chroma keying.

A silhouette image tells us that any points in the space that projects outside the silhouette are not occupied by the objects. Therefore we can limit a part of space which can be occupied by the object by intersecting every set of spatial points that project inside a silhouette, for all images. Thus this approach is also termed as volume intersection approach, and its output is known as Visual Hull [Lau94], which is a maximum volume that is consistent with the silhouettes. The volume intersection approach cannot reconstruct concavities on objects.

Another benefit of the volume intersection approach is that the computation can be highly parallelized, because space occupancy between different points in the space can be computed independently. For example, Matsuyama et al. achieved real-time dynamic 3D object shape reconstruction using PC cluster [MWTN04] based on volume intersection approach.

On the other hand, stereo approaches use texture cues on object surface. For example, Space Carving [KS00] is a provably-correct algorithm that gives a "photo hull", which is the union of all photo-consistent shape with a given set of images. It assumes object's surface radiance is "locally computable", meaning that the radiance at any point is independent of the radiance of all other points in the scene.

Most shape reconstruction methods introduce a priori knowledge about the object as the constraints to get better shape. Some algorithms assumes smoothness of the shape. It is also useful to gain robustness against noisy data. In such methods, the reconstruction problem can be formalized as a minimization problem of energy functions, that reflects both the photo-consistency and the smoothness constraint. Especially, when the smoothness is defined as a sum of local features between nearby points or voxels, use of graph-cut algorithms effectively solve such kind of energy minimization problem [SVZ00].

Some algorithms use the shape representation by a surface mesh model rather than voxel model. In contrast to the voxel representation, mesh model explicitly expresses object surfaces. It facilitates introduction of surface continuity or topology preservation [NM04], and the smoothness constraint mentioned above.

Finally, when the 3D reconstruction is applied to movie sequences, motion cues can be used as constraints to obtain better shape. Starck et al. [SH07a] fused silhouette, stereo and feature constraints. Their method minimizes the distance between the object surfaces at subsequent time frames in order to perform temporally consistent reconstruction. Tung et al. combined super-resolution technique [TNM08] to improve input image resolution. It also utilized motion cues of the object by the feature point matching between subsequent frames.

Photometric Reconstruction

Photometric property of non-transparent object surface can be represented by the Bidirectional Reflectance Distribution Function (BRDF). Tehobalt et al. [TAL+07] has proposed a system to capture reflectance characteristic changes of performer's clothing. It first estimate BRDF for each point on the performer by cap-

turing the target from varying directions under a known illumination, then fits the BRDF field to the reconstructed dynamic shape model.

Some other methods do not reconstruct BRDF but rather use surface appearance as object texture. If we assume the object surfaces to be Lambertian surfaces, a unique color for each point on the object, which represents the luminance of it, can be defined. This type of model is called "view-independent texture" [SH07b] [MTG97]. Another class of methods adopt "view-dependent" [TNYM06] texture mapping to render more plausible views. It can approximate non-Lambertian surfaces such as specular surfaces.

1.2.4 Rendering

Rendering is the process to generate a new image from 3D shape data and a viewpoint. 3D computer graphics techniques can be applied to 3D video data. However the specific problem with 3D video is imprecision of the object geometry and camera calibration. Both imprecision causes inconsistent correspondences between object surface and camera images. It results in artifacts such as blurring or ghosting in the texture mapping process. View-dependent texture mapping can improve these artifacts. For example, Floating Textures [EDDM$^+$08] and Harmonised Texture Mapping [THM10] compensates for these artifacts by the view-dependent deformation of textures from different cameras.

1.2.5 Compression

Storage and transmission of 3D video data is costly because of its large data amount.

Starck et al. [SH07a] proposed a method to control mesh density to reduce the amount of the data. The method consists of parametrization of a surface by a spherical coordinate, and then resampling the vertices. Similarly, Briceno et al. have proposed Geometry Video [BSM$^+$03] as a representation of 3D video data. Every pixel in each frame of a Geometry Video represents three-dimensional coordinate of one vertex on the mesh model. This representation can be processed in the same manner as 2D videos. For example, it facilitates the use of 2D video compression algorithms to process 3D video streams.

1.2.6 Shape and Motion Analysis

There are a number of 3D shape indexing methods for static objects [TV08]. These methods represent some features of objects in a compact manner in order to facilitate content-based 3D model retrieval or object recognition.

Several methods have been proposed to use 3D video for vision-based markerless motion capture of human performers. Some methods uses a top-down kinematic model, which defines a graph structure of the human body parts [WL01], just like marker-based optical motion capture systems. On the other hand, there are also several approaches [CJM03] [MNMM06] that extract a kinematic model of the object from a 3D video so that more general objects can be handled. Additionally, some other methods extracts object motions as deformation of a single mesh model throughout the time [dAST+08] [SH07b] [NM04]. This deformation-based representation can also deal with non-rigid object motions.

1.3 Limitations of Existing 3D Video Capture Methods

This section discusses three points of 3D video capture each of which is requisite for successful capture but limits the capture area. As described in section 1.2, 3D video generation approaches using multi-view video depends on 3D shape reconstruction algorithms. Thus the requirements on the multi-view video for 3D video generation are summarized as follows:

Req. 1 Camera calibration

Req. 2 Visual coverage

Req. 3 Spatial resolution

Requirement 1 states that the cameras must be calibrated accurately. The second requirement states that every points on the target object must be simultaneously captured from different viewpoints. Lastly, the object must be captured with high enough spatial resolution. Spatial sampling resolution by the cameras defines both reconstruction accuracy and spatial resolution of texture. 3D video can be generated in the space where these three requirements are satisfied. We call such a space “capture space”.

Existing 3D video capture systems typically use a fixed set of cameras focused on a restricted volume in order to satisfy Req. 1. In such static camera systems, reconstruction accuracy and spatial resolution of texture are effectively governed by the number of sampling elements on each camera, the focal lengths of each camera, and the distance between the object and the cameras. It means that, where the number of used cameras and their output image resolution are restricted, a trade-off problem arises between reconstruction accuracy and the size of capture space.

There are several approaches to realize 3D video capture of a moving object in a wide area. One straightforward approach is use of wide lens camera to extend each camera view. But it sacrifices spatial sampling resolution of the object surface and Req. 3 will not be satisfied. Another approach is to employ more cameras to view different places. But this approach requires numerous cameras; the number of required cameras increases in linear proportion to the size of the space where the object moves. It is not realistic from the standpoint of the cost. This approach has a problem on the effectiveness of the resource usage; most part of the cameras, processors and storage devices would be consumed to record nothing but the background in most of the time, producing no information about the object.

Another approach is use of active cameras. An active camera is a camera that can be controlled by a computer to change its view. Tracking objects with active cameras can use limited number of cameras more effectively, thus virtually extends capture space. We adopt this approach. In this thesis, we only consider pan-tilt-zoom (PTZ) cameras, which can change its viewing direction and focal length. Although we could consider active control of view locations, e.g. using a crane camera or a sliding stage, they are not considered in this thesis because such kinds of equipments are much costly but do not extend the size of capture space so much.

1.4 Difficulties with 3D Video Capture using Active Cameras

Two problems arise when we employ active cameras for 3D video capture. The first problem is camera calibration accuracy. Calibration errors directly affects 3D video reconstruction accuracy. Nonetheless, it is difficult to calibrate active cameras accurately and robustly in the presence of large changes in their focal

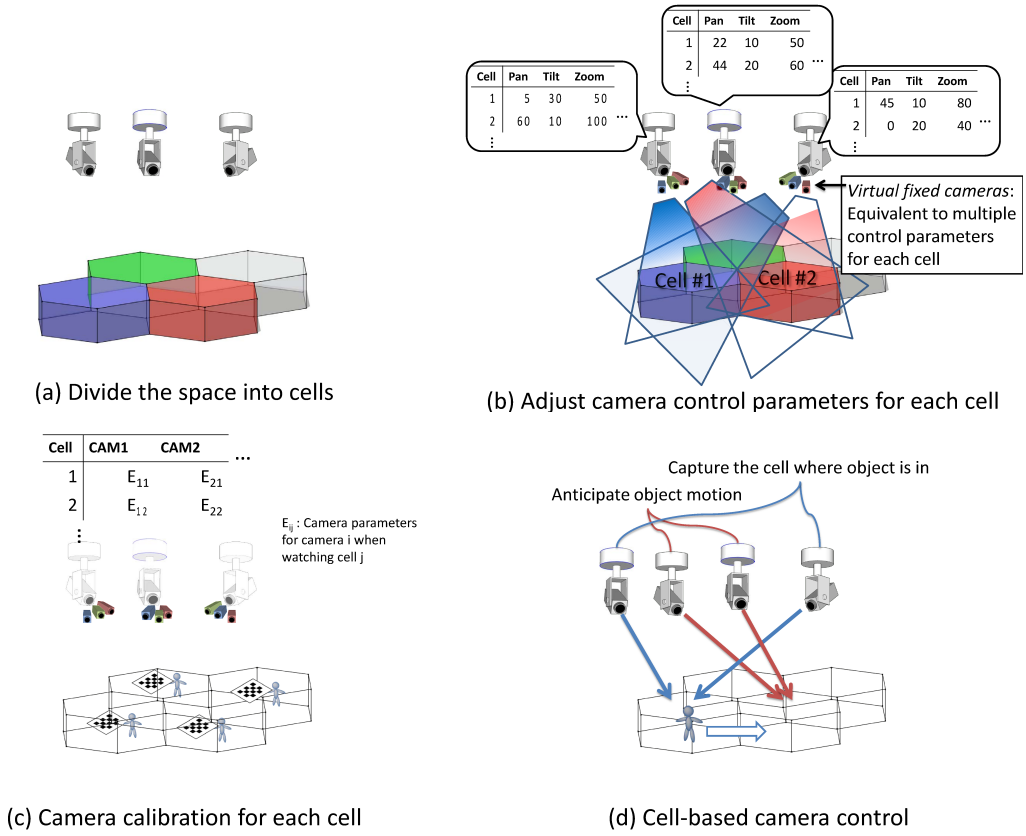


Figure 1.3: Cell-based framework.

lengths.

The other one concerns the real-time camera control for 3D video capture. 3D positions of target objects must be computed from multi-view images and the cameras must be controlled so that Reqs. 2 and 3 are satisfied in every frame.

1.5 Cell-Based Idea

In this thesis, we propose a "cell-based" concept as a basic computation framework to resolve the two difficulties stated in section 1.4. Figure 1.3 shows the general idea of our method. Our principle idea is to divide the space where the object moves into smaller subspaces named "cells".

It is straightforward to satisfy Reqs. 1 – 3 in a small space, because we can statically configure multiple cameras to view the space and calibrate the cameras, in exactly the same manner as existing 3D video capture methods using static cam-

eras. Our method perform this procedure for every cell. After that, our method controls active cameras based on cells. Each camera does not follow the object itself continuously, but chooses a cell to "watch" depending on the object position as to satisfy Req. 2 with other cameras. Finally, 3D video can be generated using the images from the cameras that have watched the object at each frame.

1.6 Outline

Chapter 2 introduces some related work on object tracking with active cameras. It introduces some existing approaches to the two difficulties described in section 1.4. Chapter 3 formulates the 3D video capture problem of a moving object in a wide area and defines our "cell-based" framework. Two different types of 3D video capture algorithms based on the framework are described in Chapters 4 and 5. In Chapter 6, A figure skater capture system is designed as an example application to show the feasibility of cell-based methods in a more realistic situation. The camera positioning problem reflecting the dynamic characteristics of active cameras is discussed. Finally, Chapter 7 concludes this thesis with discussions and possible future research directions.

Chapter 2

Related Works — Object Tracking using Active Cameras

The previous chapter stated that the goal of this thesis is to realize 3D video capture using active cameras. The chapter stated that the two main difficulties are the active camera control satisfying requirements 2 and 3 in a real time, and accurate calibration of active cameras. This chapter reviews existing approaches to these problems.

2.1 Active Tracking as Dynamic Sensor Planning

Detection, tracking, localization and identification of objects using multiple cameras are well studied in the research area of visual surveillance [SR08]. Some of such works have used multiple active cameras for wide-area surveillance, though, such works have not considered 3D reconstruction of the objects in a wide area.

As mentioned in Chapter 1, tracking objects using active cameras can realize both wide-area and high-resolution observation of objects with limited number of cameras. However, use of active cameras raises a new problem; how can "appropriate" active camera control for 3D video generation be performed automatically? From this standpoint, one of the relevant research areas is sensor planning in computer vision.

Sensor planning is defined as automatic configuration of sensors for some specific task. For example, Art Gallery Problem [O'R87] is a problem to find viewpoints that can cover entire room, when the shape of the room is given. "Con-

figuration" of sensors here includes all the variables which determine the characteristics of sensors. When we regard active cameras as sensors, we can change positions, orientations, zoom and focus motions of active cameras to change characteristics of cameras. Therefore, active camera control for object tracking can also be regarded as sensor planning problem. New camera state should be generated in every frame to track target objects. Some additional requirements are added depending on the task. In practical systems, the temporal constraints due to active camera motion should be considered as well.

There are several works that applied dynamic sensor planning, especially for visual surveillance applications. Namely, these works deal with the detection, tracking, localization and identification of objects but not 3D shape reconstruction. Spletzer et al. proposed a dynamic viewpoint planning framework for mobile robots [ST03]. It optimally tracked target objects in real time by moving robot cameras in the directions that best improve expected triangulation error of the target objects. Sommerlade et al. [SR08] proposed a scheme to find and track objects by a pan-tilt-zoom camera based on information gain maximization. Ukita et al. have proposed a multiple camera control architecture named "Cooperative Distributed Vision" [UM05]. They realized real-time detection and tracking of multiple objects by the communication between multiple visual agents, which is a computational model of a computer with an active camera connected to it. Bakhtari and Benhabib [BB07] have shown an active camera control scheme for facial recognition.

A sensor planning problem is ultimately a search problem in a large configuration space, which consists of all the possible combinations of sensor parameters. It means that finding the optimal solution is sometimes computationally difficult without good heuristics. Some of the algorithms generate several candidates and finds a solution that is optimal or that satisfies the requirement for the task. This methodology is termed "generate-and-test" approach [TAT95]. On the other hand, some other algorithms take "synthetic" approach, which directly generates the solution satisfying the requirement by some analytical method. An example of synthetic approach for viewpoint planning can be seen in the work by Tarabanis et al. [TTA95]. Their method finds good viewpoints for feature point detection when an object shape is given a priori. It computes admissible region of a viewpoint using some constraints, such as visibility constraint, and then arrange cameras in the middle of such regions to maximize its margin. They proved that such solutions are globally admissible and locally optimal.

3D video capture problem of a moving object in large area can be regarded as a sensor planning problem. All the camera parameters should be controlled as to satisfy Reqs. 2 and 3. Use of multiple active cameras causes the explosion of the state space size. Nonetheless, the camera configuration must be generated in a real time in response to the object movement. Our cell-based algorithms give pseudo-optimal solutions to this sensor planning problem.

Yous et al. proposed an assignment scheme of pan-tilt camera views for 3D shape acquisition [YUK06]. They realized object capture with high resolution using narrow-fov cameras under the condition that the object shape in the previous frame was given. Their scheme computes camera directions so that every point on object surface is observed with 2 or more cameras. It also minimize pan/tilt movement of the cameras to facilitate tracking. Though, their method did not handle zoom control or lens selection problem. Moreover, it does not have a mechanism to guarantee continuous capture of an object that moves in a wide area.

2.2 Active Camera Calibration

Most object tracking methods require camera parameters of multiple cameras for measuring object positions. Especially, when using multiple cameras to track objects cooperatively, the object position is needed for the hand-over of the object motion information between different cameras.

One straightforward approach to obtain camera parameters of active cameras is to employ some active camera model, which represents the relationship between active camera mechanics and optics, and fit such models to actual cameras. For instance, partially-fixed viewpoint pan-tilt-zoom (PFV-PTZ) camera [KWM03] [Mat98] model is one of the practical models. Some works adopted more complex active camera model [JKKW06]. Using such models, camera parameters can be computed from pan, tilt and zoom positions of the cameras at each frame. One example multi-view video capture system that took this approach has been shown by Kitahara et al. [KSA+01]. They employed a set of dedicated, specially designed active cameras which can synchronously measure pan, tilt, zoom and focus positions of every camera. They realized active tracking and 3D shape reconstruction of an object using the system. However, this class of approaches has a difficulty in its accuracy, especially when applied to off-the-shelf active cameras. Firstly, it is difficult to build an accurate camera model that

suits actual active camera, especially in the presence of large changes in the focal length. Secondly, not all off-the-shelf active cameras provide synchronous capture of pan/tilt/zoom positions to their exposure timing. This limitation makes it difficult to obtain accurate camera parameters when a camera continuously changes its view to follow objects.

Another approach is to estimate camera parameters from captured videos themselves. If accurate positions of static landmarks are obtained in advance and they are identified in the images, the orientations of cameras can be estimated. It requires that enough number of landmarks are accurately detected and located from each camera. Therefore the view angle of each camera should be set wide enough. However, it conflicts with Req. 3. Additionally, the landmarks on the background are often occluded by the object. Moreover, the focal lengths of the cameras varies in time. It makes the problem even more difficult.

The cell-based framework takes none of these approaches. Thus the problems on active camera calibration can be avoided. Cell-based calibration can achieve accurate camera calibration to the same level with that for static cameras.

Chapter 3

Cell-Based Framework for 3D Video Capture Algorithms

Our "Cell-Based" framework itself does not define any specific algorithm, but rather defines a computational framework on which several algorithms can be based. This chapter first formulates active camera control problem for 3D video capture, then describes the cell-based framework as a solution, finally declares algorithm design considerations when building algorithms based on the framework.

3.1 3D Video Capture Problem Formulation

As mentioned in Chapter 1, the requirements for 3D video generation from multi-view video are summarized as follows.

Req. 1 Camera calibration. All the active cameras must be calibrated accurately.

Req. 2 Visual coverage. Every point on the object surface must be observed from multiple viewpoints.

Req. 3 Spatial resolution. Every point on the object surface must be observed with high spatial resolution.

3D video capture using active cameras also requires real-time camera control satisfying these requirements. Consequently, we derive the fourth requirement:

Req. 4 Track target objects in real time satisfying Reqs. 2 and 3.

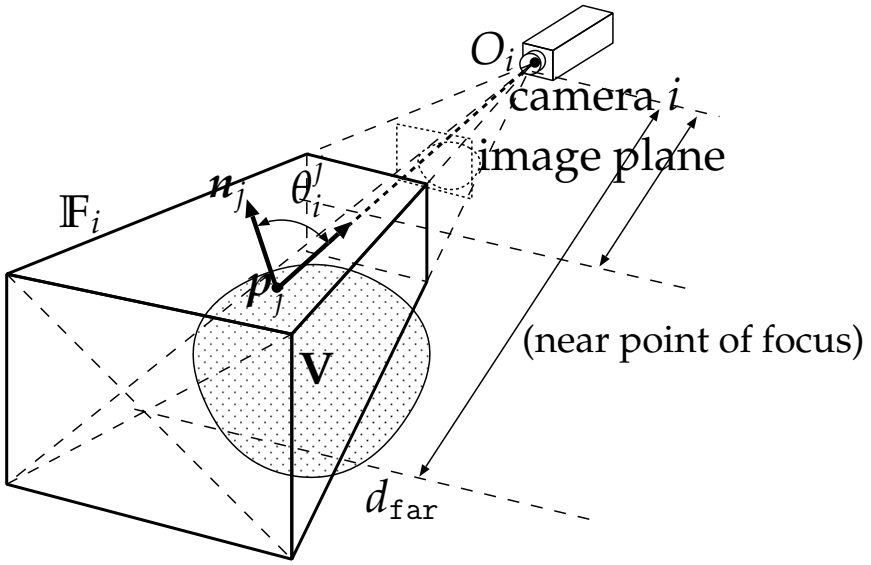


Figure 3.1: Geometric relations of the symbols that define the visual coverage ©2010 Springer [YYM10]

In our cell-based framework, Req. 1 can be satisfied by camera calibration on each cell. Req. 3 can be satisfied by adjusting zooms for each cells. Req. 4 can be satisfied by on-line tracking and camera control based on cells. In contrast, considering Req. 2, complete visual coverage cannot be always satisfied in 3D video generation from multi-view video. This is because the visual coverage depends on the shape of captured objects, as well as the relative position of objects to cameras. In extreme cases, some types of object shape hinder complete visual coverage by self-occlusion. Additionally, there is a trade-off problem between Req. 2 and Req. 3. Narrower angle of view is preferred from the standpoint of Req. 3, however, zooming in on a small part of the object fosters observing other points on the object. It affects satisfaction of Req. 2. In this thesis, we give higher priority to Req. 2; our method guarantees the spatial resolution to be higher than the lowest allowable resolution, but no longer optimizes the active camera control for the resolution. For this purpose, we first define an evaluation function that quantifies how well Req. 2 is satisfied. Then we formulate camera control problem for 3D video capture as an optimization problem of camera control values.

We define the symbols to formulate the problem in Table 3.1. Figure 3.1 shows the geometric relations of some symbols.

Table 3.1: Notations for defining the Visual Coverage

$\mathbb{S} \subset \mathbb{R}^3$	Target space; the space where the object can move.
$N_{\mathtt{cam}}$	Number of active cameras to be used.
$\boldsymbol{E}_i (i = 1, \cdots, N_{\mathtt{cam}})$	Camera parameters; parameter values of certain camera projection model adopted by the capture system. Includes intrinsic and extrinsic, geometry and photometry parameters.
s [mm/pixel]	Lowest allowable spatial resolution.
$\mathbb{F}_i$	View frustum: The space where objects can be captured by camera i with higher resolution than s. This is bounded by the distance from the camera, the depth of field, and the angle of view.
$\mathbf{V} \subset \mathbb{S}$	Part of the space occupied by the object.
$\vec{p}_j (j = 1, 2, \ldots, N_{\mathtt{p}})$	Points on the object surface.
$\vec{n}_j$	Surface normal vectors on $\vec{p}_j$.
O_i	Projection center of active camera i.
θ_i^j	Emission angle of the ray from $\vec{p}_j$ to active camera i; angle formed by $\vec{n}_j$ and $(O_i - \vec{p}_j)$.
$\mathtt{visible}(\mathbf{V}, \vec{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i))$	Visibility function: Binary function that returns 1 when a point $\vec{p}_j$ on the surface of the object $\mathbf{V}$ is visible by a camera i with state $(\boldsymbol{E}_i, \mathbb{F}_i)$ and 0 when unobservable.

3.1.1 Active Camera Model

In this thesis, we use pan-tilt-zoom (PTZ) cameras as active cameras. It means that we can model active cameras as the cameras that change directions and focal lengths according to each camera control value, which consists of pan, tilt, zoom and focus positions. Since these parameter changes accompany physical motions of cameras, significant time lags exist between the transmission of these parameters and the end of corresponding motions. After the camera control values are fixed and the camera motion is finished, each camera's projection process is modeled in the same manner as static cameras, e.g. a pinhole camera model with radial lens distortion [Zha00].

Based on these assumptions, we first formulate Requirements 2 and 3 for a static object and cameras in the next two sections. After that, 3D video capture of dynamic object using active cameras is formulated.

3.1.2 Spatial Resolution

Requirement 3 states that object surface must be observed with high spatial resolution. We define spatial resolution as the minimum distance between two nearest distinguishable points on the object by a camera. Our method guarantees that every point on the object is observed with higher resolution than lowest-allowable resolution given by the user.

The spatial resolution is governed by the defocusing and the perspective projection process. Perspective projection is governed by the focal length of the camera, number of sampling elements in the camera and the distance to the object from the camera. When the object is focused sharply enough, number of sampling elements is the limiting factor for the spatial resolution. For example, assume that the object is at d millimeters apart from the projection center of a camera, the camera has W pixels in its image row, and its horizontal viewing angle is θ_{h} radians. Two points on the object must be

$$s = \frac{2d}{W} \tan \theta_{\mathrm{h}} \tag{3.1}$$

millimeters apart from each other in order to be distinguishable in the image. Inversely, when the lowest allowable resolution is given as s[mm/pixel], any object surface farther than

$$d_{\mathtt{far}} = \frac{sW}{2 \tan \theta_{\mathrm{h}}} \tag{3.2}$$

cannot be observed with finer resolution than s[mm/pixel] by the camera. The same discussion can be held for the vertical direction of the image. Though, it is usually the same as the horizontal one described above, because each pixel of the camera can be regarded as square. By this reason, we adopt the value of Eq. (3.1) as the definition of spatial resolution.

We define view frustum of each camera — region of the space that can be captured by camera i with higher resolution than s — in order to express the spatial resolution requirement effectively. Each view frustum is modeled as a rectangular frustum. The distance to the near plane is governed by the near point of focus. On the other hand, the distance to the far plane is governed by both the far point of focus and the distance limit where the spatial resolution becomes lowest allowable, as shown in Fig. 3.1. The shorter one governs the distance to far plane.

3.1.3 Visual Coverage

Assume that $N_{\mathtt{cam}}$ cameras with camera parameters $\boldsymbol{E}_i$ are viewing an object occupying a volume $\mathbf{V} \subset \mathbb{S}$. Let $\{\vec{p}_j\}_{j=1,2,\ldots,N_{\mathrm{p}}}$ be the points on the object surface, and $\vec{n}_j$ be the surface normal vectors on the jth point, respectively. In order to generate a 3D video, the following two conditions must be satisfied for each point $\vec{p}_j$.

- $\vec{p}_j$ must be observed from the viewpoint directly facing the surface in order to get the texture information nearby.
- $\vec{p}_j$ must be observed from the orthogonal direction to the surface normal in order to get better initial shape with Shape-From-Silhouette.

As mentioned, it is sometimes impossible to satisfy these conditions for all of the surface points, considering the shape complexity or self-occlusion of target objects. We first quantify these two conditions for a given pair of point $\vec{p}_j$ and camera i with parameter $\boldsymbol{E}_i$ and view frustum $\mathbb{F}_i$.

$$q_{\mathtt{t}}(\mathbf{V}, \vec{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i)) = \mathtt{visible}(\mathbf{V}, \vec{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i)) \cos\theta_i^j \tag{3.3}$$

$$q_{\mathtt{s}}(\mathbf{V}, \vec{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i)) = \mathtt{visible}(\mathbf{V}, \vec{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i)) |\sin\theta_i^j| \tag{3.4}$$

The larger the values of these functions are, the more each requirement is satisfied. These functions imply that it is impossible to satisfy the two requirements by a

single camera. However, each requirement can be satisfied by different cameras in the 3D video production with multiple cameras. Thus we quantify the two requirements on each point $\vec{p}_j$ as follows:

$$q'_{\mathtt{t}}(\mathbf{V}, \vec{p}_j, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) = \max_i q_{\mathtt{t}}(\mathbf{V}, \vec{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i))$$
$$q'_{\mathtt{s}}(\mathbf{V}, \vec{p}_j, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) = \max_i q_{\mathtt{s}}(\mathbf{V}, \vec{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i))$$

Then, we choose the least observable point by Eqs. (3.5) and (3.6).

$$Q_{\mathtt{t}}(\mathbf{V}, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) = \min_j q'_{\mathtt{t}}(\mathbf{V}, \vec{p}_j, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) \tag{3.5}$$
$$Q_{\mathtt{s}}(\mathbf{V}, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) = \min_j q'_{\mathtt{s}}(\mathbf{V}, \vec{p}_j, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) \tag{3.6}$$

The larger the values of Eqs. (3.5) and (3.6) are, the lager area on the object is observed well. Finally, we define an evaluation function that estimates how well Req. 2 is satisfied by Eq. (3.7).

$$Q(\mathbf{V}, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) = Q_{\mathtt{t}}(\mathbf{V}, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) Q_{\mathtt{s}}(\mathbf{V}, \{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}) \tag{3.7}$$

Using Eq. (3.7), Req. 2 can be quantitatively evaluated for a given pair of $\mathbf{V}$ and $\{(\boldsymbol{E}_i, \mathbb{F}_i)\}_{i=1,\cdots,N_{\mathtt{cam}}}$. This is our definition of the visual coverage for a static scene and the cameras.

3.1.4 3D Video Capture using Active Cameras

Let $\mathbb{V}(t) \subset \mathbb{S}$ be the space occupied by the object at time t in order to deal with the object movements. The active cameras are controlled via camera control values $\boldsymbol{e}_i(t)$ at each time t. Therefore, we denote the camera parameters and view frustums at each time as the functions of $\boldsymbol{e}_i(t)$; $\hat{\mathbf{E}}_i(\boldsymbol{e}_i(t))$ and $\hat{\mathbb{F}}_i(\boldsymbol{e}_i(t))$. Figure 3.2 and Table 3.2 summarizes the notations.

If $\{\boldsymbol{e}_i(t)\}_{i=1,2,\cdots,N_{\mathtt{cam}}}$ were independent to each other, the process to capture the whole object surface can be formulated as follows: when given $\mathbb{V}(t)$, optimize $\{\boldsymbol{e}_i(t)\}_{i=1,2,\cdots,N_{\mathtt{cam}}}$ for maximizing $Q(\mathbb{V}(t), \{\hat{\mathbf{E}}_i(\boldsymbol{e}_i(t)), \hat{\mathbb{F}}_i(\boldsymbol{e}_i(t))\}_{i=1,\cdots,N_{\mathtt{cam}}})$ for every t. However, it is not the case in practical systems. $\{\boldsymbol{e}_i(t)\}_{i=1,2,\cdots,N_{\mathtt{cam}}}$ are under constraints of camera motion dynamics. Thus the optimization problem must be solved for all t simultaneously. Accordingly, the target functions must be aggre-

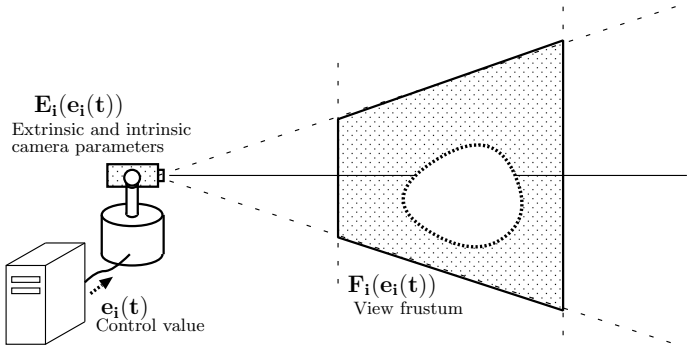


Figure 3.2: Our active camera model. Each camera receives camera control value and changes its direction, focal length and focusing. View frustum is uniquely determined by a control value.

Table 3.2: Notations for defining the 3D video capture problem using active cameras

$\boldsymbol{e}_i(t)$	Camera control value for camera i at time t.
$\hat{\mathbf{E}}_i(\boldsymbol{e}_i(t))$	Camera parameters of camera i with control value $\boldsymbol{e}_i(t)$.
$\hat{\mathbb{F}}_i(\boldsymbol{e}_i(t))$	View frustum of camera i with control value $\boldsymbol{e}_i(t)$
$\mathbb{V}(t) \subset \mathbb{S}$	Part of the space occupied by the object at time t.
$\mathbf{D}$	Domain of the object movement, within which the object moves.
$\vec{x} \in \mathbf{D}$	Object position.
$\tilde{\mathbb{V}}(\vec{x}) \subset \mathbb{S}$	Bounding volume of the object when the object is located at x. The object can change its shape to any form as long as it resides inside the bounding volume located at x. Namely, $(\mathbb{V}(t) \subset \tilde{\mathbb{V}}(\vec{x}(t)))$ should be satisfied for all $\vec{x}(t)$.

gated into a single scalar function as well so that the "optimal" camera control can be defined uniquely. In this thesis, we attempt to ensure that 3D video capture is performed without frame drops or degraded quality at certain frames. For this purpose, we define the overall visual coverage for the whole sequence by the visual coverage at the worst frame.

$$\min_t Q(\mathbb{V}(t), \{\hat{\mathbf{E}}_i(\boldsymbol{e}_i(t)), \hat{\mathbb{F}}_i(\boldsymbol{e}_i(t))\}_{i=1,\cdots,N_{\mathtt{cam}}}) \tag{3.8}$$

Finally, 3D video capture problem is formulated as follows: optimize $\{\boldsymbol{e}_i(t)\}_{i=1,2,\cdots,N_{\mathtt{cam}}}$ for maximizing Eq. (3.8).

3.1.5 Object Model

The optimization problem described above is unsolvable in its original form. One of the reasons is that we cannot know $\mathbb{V}(t)$ before 3D video capture because it is the object shape itself.

The scope of this thesis is 3D video capture of an object that moves in a large area. It means that the object may move within a large space but the object is relatively small to the space size. Therefore we can consider the shape and position of the object separately. For this reason, we represent the object's position at time t by a single point $\vec{x}(t) \in \mathbf{D}$, where $\mathbf{D}$ is the domain of the object locomotion. For example, if the object were to move freely in the space, $\mathbf{D}$ equals to $\mathbb{S}$. If the object were assumed to move on the floor, $\mathbf{D}$ can be a two-dimensional subspace; e.g. the floor plane. In this case we would adopt the projection point of the object centroid to the floor plane as $\vec{x}(t)$.

As for the object deformation, we assume that the object can be contained in a limited volume nearby $\vec{x}(t)$.

$\tilde{\mathbb{V}}(\vec{x}) \subset \mathbb{S}$ Bounding volume of the object when the object is located at x. The object can change its shape to any form as long as it resides inside the bounding volume located at x. Namely, $(\mathbb{V}(t) \subset \tilde{\mathbb{V}}(\vec{x}(t)))$ should be satisfied for all $\vec{x}(t)$.

Cell-based algorithms require $\tilde{\mathbb{V}}(\vec{x})$ as its input. Hereafter we do not need to consider object deformation. Cell-based algorithms control cameras solely based on the object position.

3.2 Cell-Based 3D Video Capture Framework

Another reason for the optimization problem for Req. 2 to be unsolvable in its original form, is its computational complexity. There are numerous possible combinations of values for $\boldsymbol{e}_i(t)$. Nonetheless Eq. (3.8) cannot be computed without fixing all the camera control values. This characteristic of the function makes it difficult to apply efficient algorithms such as dynamic programming.

Our cell-based framework performs pseudo-optimal camera control by the steps listed below:

1. Studio Setup
2. Cell Generation
3. View Adjustment
4. Cell-based Calibration
5. Cell-based Camera Control
6. 3D video generation

The cell-based framework separates the camera control problem into spatial planning and temporal planning. Steps 1 through 4 ensures to satisfy Reqs. 1 and 3 in each cell. Thus the 3D video reconstruction is ensured as long as the object stays inside one of the cells. Since the object moves from a cell to another, each camera must change its views in order to track the object. During this process, Req. 2 must be satisfied continuously on the moving object; otherwise, some of the 3D video frames would be lost. Such "hand-off" between the cells is conducted in step 5.

3.2.1 Studio Setup

Sets up multiple active cameras around the target area where the object moves. Since we only consider PTZ cameras, this step determines the layout of the cameras.

3.2.2 Cell Generation

Divides the space into cells. We define a cell as a topological subspace of $\mathbf{D}$.

$$\mathbf{C}_j \subset \mathbf{D} \tag{3.9}$$

This step outputs multiple cells, $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_{N_{\mathtt{cell}}}$. Number of the cells $N_{\mathtt{cell}}$ is also determined in this step. The union of all cells should cover entire space.

$$\bigcup_{j=1,2,\ldots,N_{\mathtt{cell}}} \mathbf{C}_j = \mathbf{D} \tag{3.10}$$

Cells are basically disjoint to each other, but not limited to it.

3.2.3 View Adjustment

Precomputes camera control values for each cell. Each camera view is adjusted so as to satisfy Req. 3 in each cell. This process outputs $\hat{\boldsymbol{e}}_i^j$ for all combinations of i and j, where $i = 1, 2, \ldots, N_{\mathtt{cam}}$ and $j = 1, 2, \ldots, N_{\mathtt{cell}}$.

3.2.4 Cell-based Calibration

Calibrate cameras for every cell in order to satisfy Req. 1 in every cell. Namely, $\mathbf{E}_i(\hat{\boldsymbol{e}}_i^j)$ are obtained for $i = 1, 2, \ldots, N_{\mathtt{cam}}, j = 1, 2, \ldots, N_{\mathtt{cell}}$. All the active cameras can be regarded as fixed cameras, while they watch one of the cells. Therefore, any existing camera calibration methods for the virtual fixed cameras, such as Zhang's[Zha00] and Svoboda's[SMP05], can be applied. Since this is an off-line process separated from the next object capture process, we can use any calibration targets such as a planar calibration chart or a point light source to obtain camera parameters accurately and robustly.

3.2.5 Cell-based Camera Control

Controls cameras based on the cells as to satisfy Reqs. 2 and 4. This step also stores all the pairs of camera images and camera state. Camera state here means whether the camera was in motion or not, and if not in motion, which cell the camera watched.

3.2.6 3D video generation

Generates 3D video from the images and the camera parameters. Camera parameters in each frame can be obtained by combining the result of the cell-based calibration and the camera state record by the cell-based camera control. We do not use images from in-motion cameras for 3D video generation since they are not calibrated.

3.3 Algorithm Design Considerations

3.3.1 Hand-off Strategy

The framework does not define how steps 2 and 5 are performed since these steps are two major design decisions to be made when building an algorithm based on the framework. These two steps concerns the hand-off strategy. Different strategies are tailored in Chapters 4 and 5 reflecting the object locomotion assumptions in each problem.

Cell Shape and Arrangement (at Step 2) The size, shape and arrangement of the cells. The cells should cover the entire space where the object may exist.

Real-time Camera Control (at Step 5) The algorithm should decide when and where each camera should be directed to, so that Req. 2 is satisfied.

3.3.2 Camera Layout

As implied by Eq. (3.3) and (3.4), the visual coverage is subject to change depending on the relative positions of the cameras to the object. We use PTZ cameras as declared in Chapter 1. It means that the positions of the cameras cannot be controlled actively. Instead the layout of the cameras should be fixed first.

As for this problem, we basically follow the solution that has been taken in existing 3D video studio design methodology. In existing 3D video studios, multiple cameras surround a target area where the object moves, and are equally distributed in all the directions. Since the object shape is not given in advance to capture, this is one reasonable solution to maintain the shape reconstruction equally well over the entire object surface.

By this reason, the algorithms in Chapters 4 and 5 assume that enough number of cameras are already set up to capture the object in the target area from varying

directions. The camera layout problem is not the main focus of this thesis, though, Chapter 6 discusses part of the camera layout problem specific to our cell-based framework.

3.4 Summary

3D video capture using active cameras can be formulated as an optimization problem of the camera control values. But the problem is unsolvable in its original form. The cell-based 3D video capture framework gives a pseudo-optimal solution to this problem. The framework consists of multiple camera studio setup, division of the space into cells, adjustment of camera control parameters for each cell, cell-based calibration, and the camera control based on the cells. The specific algorithms for cell generation and camera control should be designed reflecting the object locomotion assumptions.

Chapter 4

Cell-Based 3D Video Capture of an Object Moving along a Given Path

This chapter proposes a cell-based 3D video capture algorithm for a single object that moves along a given path in one direction. Since the object motion is restricted to the path, the algorithm partitions the path into a set of cells. Object tracking for 3D video capture is conducted by two steps; firstly, camera control schedule for all the cameras is built by a pseudo-optimal assignment of "camera modes" to each point of the path. Secondly, the cameras are controlled in a real time using the schedule. This chapter also shows experimental results and performance evaluation of the algorithm.

4.1 Problem Description

The scope of the problem in this chapter is summarized as follows:

- There is only one target object.
- The object moves along a given path in one direction.
- Maximum velocity of the object is given.
- The resolution requirement is specified by the lowest allowable resolution.
- The cameras are active PTZ cameras. A camera control value consists of pan, tilt, zoom and focus, and their projection centers are almost fixed.
- The cameras are surrounding the space where the object moves to view the object there from varying directions.

4.2 Approach

We design our algorithm using the cell-based framework described in Chapter 3. Two main ideas in the algorithm is as follows:

Shape and Arrangement of Cells Since the object moves along a given path, the position of the object can be represented by arc length along the path. It means that the domain of the object locomotion, which has been introduced in Chapter 3, is the path itself. Therefore the algorithm divides the path into cells.

Real-time Active Camera Control The algorithm first builds a camera control schedule in an off-line process, then tracks the object using the schedule. The schedule consists of the assignment of three roles for each camera at every point on the given path: watching the cell where the object is in, switching its view to the next cell, or watching the next cell to anticipate the object movement. The algorithm generates a pseudo-optimal schedule from the standpoint of Req. 2.

4.3 Algorithm

The inputs to our algorithm consists of scenario and resources described in Table 4.1 and 4.2, and the output is a 3D video of a moving object in a widespread area. The algorithm consists of five processes.

1. Cell formation
2. Camera calibration
3. Camera control scheduling
4. Real-time object tracking and camera control
5. 3D video generation

We use the symbols listed in Tables 4.3 and 4.4 in the following descriptions.

Table 4.1: Scenario and its notations

$\{\mathbb{L}(l) \in \mathbb{S} \mid l \in [0, L]\}$	Path of the object motion to capture, expressed as a curve with length L, and parametrized by arc length. The whole curve is denoted by $\mathbb{L}$ for simplicity.
L	Length of the path.
$\tilde{\mathbb{V}}(\vec{x}) \subset \mathbb{S}$	Bounding volume of the object. When the object is at $\vec{x}(t)$, $\mathbb{V}(t) \subset \tilde{\mathbb{V}}(\vec{x}(t))$ must be satisfied.
$v_{\mathtt{max}}$	Maximum allowable velocity of the object movement.
s	Lowest allowable spatial resolution.

Table 4.2: Resources and its notations

$\tau_{\mathtt{cam}}$	Video capture interval.
$K_i(\mathbf{V})$	Function that computes a camera control value $\boldsymbol{e}_i$ for camera i such that all the points in $\mathbf{V}$ is included in the camera's view frustum defined in Section 3.1.2. If there is no such $\boldsymbol{e}_i$, $K_i(\mathbf{V}) = \phi$. This function must be designed reflecting the structure of the active camera.
$\tau_i(\boldsymbol{e}_i, \boldsymbol{e}_i')$	Length of time needed for changing the state of camera i from $\boldsymbol{e}_i$ into $\boldsymbol{e}_i'$ and getting the first image from the camera with state $\boldsymbol{e}_i'$.
$\tau_{\mathtt{proc}}$	Length of time needed to measure the 3D position of the object.
$\mathtt{visible}(\mathbf{V}, \boldsymbol{p}_j, (\boldsymbol{E}_i, \mathbb{F}_i))$	Visibility function (cf. Section 3.1.) This must be given depending on the studio setup.

Table 4.3: Symbols for the cell formation algorithm

$\Delta l = v_{\mathtt{max}}\tau_{\mathtt{cam}}$	Length of each path fragment.
$\mathbf{f}(n)(n = 0, \ldots, N_{\mathtt{f}} - 1)$	Path fragments.
$N_{\mathtt{f}}$	Total number of fragments generated by the algorithm.
$\mathbb{W}(n)$	Unit space: space along the nth fragment.
$\mathbf{C}_k \subset \mathbb{L}(k = 1, \ldots, N_{\mathtt{cell}})$	Cells.
$N_{\mathtt{cell}}$	Total number of the cells generated by the algorithm.
c_k	Cell border fragment indices. Fragments from $\mathbf{f}(c_k)$ to $\mathbf{f}(c_{k+1})$ belongs to cell k.
$\hat{\boldsymbol{e}}_i^k$	Camera control value for camera i to watch the kth cell.

Table 4.4: Symbols for camera control scheduling and real-time tracking

m_i^n	Camera mode. The index of the cell that camera i should watch when the object is on fragment n.
$g^n \subset \{1, 2, \ldots, N_{\mathtt{cam}}\}$	Subset of cameras that watches a cell when the object is at fragment n
A_k	Fragment indices where the scheduling problem is divided.
$\vec{x}(t)$	Object position.
$\boldsymbol{I}_i(t)$	Image captured by camera i at time t.

4.3.1 Cell Formation

This subsection first describes the definition of cells, conditions that cells should satisfy, and then states the cell formation algorithm.

The object moves on the path given by the scenario. Therefore the cell formation algorithm divides the path into cells. Namely, each cell is a topological subspace of $\mathbb{L}$.

As mentioned in Section 3.1, our method satisfies Req. 1 and 3 by cell-based processes. By the definition of view frustum in Chapter 3, Req. 3 within a single cell $\mathbf{C}_k$ can be expressed by:

$$\forall \vec{x} \in \mathbf{C}_k, \tilde{\mathbb{V}}(\vec{x}) \subset \hat{\mathbb{F}}_i(\hat{\boldsymbol{e}}_i^k), \tag{4.1}$$

which is equivalent to

$$\bigcup_{\vec{x} \in \mathbf{C}_k} \tilde{\mathbb{V}}(\vec{x}) \subset \hat{\mathbb{F}}_i(\hat{\boldsymbol{e}}_i^k). \tag{4.2}$$

We call the left hand side of (4.2) as "the common view volume of $\mathbf{C}_k$." Since the view frustum of each camera is limited, the size of each cell is limited by Eq. (4.2). On the other hand, considering the camera calibration costs, the number of the cells should be minimized. In other words, each single cell should be as large as possible.

The cell formation algorithm divides the path into cells that satisfy the conditions above. First, we discretize the given path into fragments. The length of each fragment is $\Delta l = v_{\mathtt{max}} \tau_{\mathtt{cam}}$, which is the maximum possible length that the object can advance in one video capture interval. We also define the unit spaces as the parts of the studio space that may be occupied by the object when the object is on each associated fragment. Eqs. 4.4 and 4.5 show the definitions of the fragments and the unit spaces respectively.

$$N_{\mathtt{f}} = \lceil \frac{L}{\Delta l} \rceil \tag{4.3}$$

$$\mathbf{f}(n) = \{\mathbb{L}(l) | l \in [\Delta l n, \max(L, \Delta l(n+1))]\} (n = 0, \ldots, N_{\mathtt{f}} - 1) \tag{4.4}$$

$$\mathbb{W}(n) = \bigcup_{\vec{x} \in \mathbf{f}(n)} \tilde{\mathbb{V}}(\vec{x}) \tag{4.5}$$

Second, we define each cell $\mathbf{C}_k$ as a union of some consecutive $\mathbf{f}(n)$ by the cell formation algorithm presented in Fig. 4.1. This algorithm outputs $N_{\mathtt{cell}}$, a list of $N_{\mathtt{cell}}$ cells and camera control values for each cell that satisfies Eq. (4.2). When

$k \leftarrow 1$
$c_k \leftarrow 0$
while $c_k < N_{\mathtt{f}} - 1$ **do**
  for i = 1,2,...,$N_{\mathtt{cam}}$ **do**
    $n_i \leftarrow c_k$
    while $n_i < N_{\mathtt{f}} - 1$ and $K_i \left(\bigcup_{n=c_k}^{n_i+1} \mathbb{W}(n) \right) \neq \phi$ **do**
      $n_i \leftarrow n_i + 1$
    end while
    if $n_i = c_k$ **then**
      return Φ
    end if
    $\hat{\boldsymbol{e}}_i^k \leftarrow K_i \left(\bigcup_{n=c_k}^{n_i} \mathbb{W}(n) \right)$
  end for
  $c_{k+1} \leftarrow \min\{n_i\}_{i=1,2,\ldots,N_{\mathtt{cam}}}$
  $\mathbf{C}_k \leftarrow \bigcup_{n=c_k}^{c_{k+1}} \mathbf{f}(n)$
  $k \leftarrow k + 1$
end while
$N_{\mathtt{cell}} \leftarrow k - 1$
return $\{N_{\mathtt{cell}}, \{c_k\}_{k=1,2,\ldots,N_k+1}, \{\mathbf{C}_k\}_{k=1,2,\ldots,N_k}, \{\hat{\boldsymbol{e}}_i^k\}_{i=1,\cdots,N_{\mathtt{cam}},k=1,2,\ldots,N_k}\}$

Figure 4.1: Cell formation algorithm ©2009 IEICE [YYMM09]

there is no such solution, it returns the empty list Φ, meaning that our method cannot satisfy Req. 3 for given scenario with given resources. In this case, our algorithm terminates at this step.

Note that a camera watching $\mathbf{C}_k$ can also satisfy Reqs. 1 and 3 in some part of neighbor cells i.e. $\mathbf{C}_{k-1}$ and $\mathbf{C}_{k+1}$. These parts of camera views are not necessary for 3D video capture in $\mathbf{C}_k$, but we can make use of it in the camera control scheduling algorithm. As described in Section 4.3.3, such spatial redundancy of camera views are useful for improving the visual coverage.

4.3.2 Camera Calibration

In order to satisfy Req. 1, camera parameters for each cell, $\hat{\mathbf{E}}_i(\hat{\boldsymbol{e}}_i^k)$, are obtained by camera calibration. All the active cameras can be regarded as fixed cameras, when they are watching one of the cells. Thus any existing camera calibration methods for fixed cameras, e.g. Zhang's [Zha00] and Svoboda's [SMP05], can be applied here.

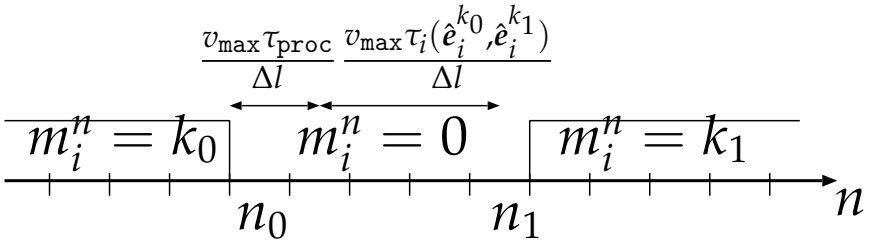


Figure 4.2: Assignment of m_i^n reflecting $\tau_i(\vec{e}, \vec{e}')$ and $\tau_{\mathtt{proc}}$ ©2009 IEICE [YYMM09]

4.3.3 Camera Control Scheduling

In order to satisfy Req. 2, camera control values that maximize the evaluation function Eq. (3.8) are needed. Our algorithm performs this computation by path fragments.

We introduce camera modes m_i^n that express the active camera states. Camera mode for camera i at nth fragment is defined as:

$$m_i^n = \begin{cases} k & \text{watching } \mathbf{C}_k \text{ with parameter } \hat{\boldsymbol{e}}_i^k \\ 0 & \text{switching its view} \end{cases}$$

These camera modes have the following constraint: Assume that camera i begins to switch its view from $\mathbf{C}_{k_0}$ to $\mathbf{C}_{k_1}$ when the object arrives at $\mathbf{f}(n_0)$. There are two kinds of delays before the camera finishes switching its view. The first one is the processing time $\tau_{\mathtt{proc}}$, which includes capturing an image, computing the object position, and communicating between computers and active cameras. After that, the active camera requires $\tau_i(\hat{\boldsymbol{e}}_i^{k_0}, \hat{\boldsymbol{e}}_i^{k_1})$ time before finishing its motion and resume capturing $\mathbf{C}_{k_1}$. Thus, this camera movement requires $\Delta T = \tau_{\mathtt{proc}} + \tau_i(\hat{\boldsymbol{e}}_i^{k_0}, \hat{\boldsymbol{e}}_i^{k_1})$ in total. Meanwhile, the object can advance by $v_{\max}\Delta T$ at the worst case. As shown in Fig. 4.2, the camera is not guaranteed to capture the object when the object is within the fragments, from $\mathbf{f}(n_0)$ to $\mathbf{f}(n_1 = n_0 + \lfloor \frac{v_{\max}\Delta T}{\Delta l} \rfloor)$. As a result, $m_i^{n_0} = 0, \cdots, m_i^{n_1} = 0$ must be assigned.

Based on the camera modes, we first consider the case when the object is within one of the path fragments $\mathbf{f}(n)$ at time t_0. Only the subset of cameras that is watching a cell,

$$g^n = \{i | m_i^n \neq 0\}$$

can capture the object, satisfying Req. 1. Note that when the object is in $\mathbf{C}_k$, a camera watching another cell $\mathbf{C}_{k'}(k' \neq k)$ can sometimes capture the object as

mentioned in Section 4.3.1, therefore such cameras are contained in g^n. We compute the evaluation function for Req. 2, firstly introduced as Eq. (3.7) in Chapter 3, using only the cameras that belong to g^n as follows:

$$Q\left(\mathbb{V}(t_0), \{(\hat{\mathbf{E}}_i(\hat{\boldsymbol{e}}_i^{m_i^n}), \hat{\mathbb{F}}_i(\hat{\boldsymbol{e}}_i^{m_i^n}))|i \in g^n\}\right) \tag{4.6}$$

Next, from Eq. (4.5) and the definition of $\tilde{\mathbb{V}}$, $\mathbb{V}(t_0) \subset \mathbb{W}(n)$ can be derived. Therefore we assume that $\{\hat{\boldsymbol{e}}_i^{m_i^n}|i \in g^n\}$ which maximizes the evaluation function for $\mathbb{W}(n)$ also maximizes the evaluation function for $\mathbb{V}(t_0)$ as well. Thus we compute the evaluation function for visual coverage for each fragment n using Eq. (4.7).

$$Q(\mathbb{W}(n), \{(\hat{\mathbf{E}}_i(\hat{\boldsymbol{e}}_i^{m_i^n}), \hat{\mathbb{F}}_i(\hat{\boldsymbol{e}}_i^{m_i^n}))|i \in g^n\}) \tag{4.7}$$

Finally, from the standpoint of guaranteeing continuous 3D video capture, we choose the worst value of Eq. 4.7 in the path as the objective function. In conclusion, the scheduling algorithm solves the following maximization problem.

Cell-based Camera Control Scheduling Problem

Variables $\{m_i^n\}_{i=1,\cdots,N_{\mathtt{cam}},n=0,\cdots,N_{\mathtt{f}}}$

Objective Function

$$\min_{n=0,\ldots,N_{\mathtt{f}}-1} Q(\mathbb{W}(n), \{(\hat{\mathbf{E}}_i(\hat{\boldsymbol{e}}_i^{m_i^n}), \hat{\mathbb{F}}_i(\hat{\boldsymbol{e}}_i^{m_i^n}))|i \in g^n\}) \tag{4.8}$$

Constraints If the gaze of camera i is switched from $\mathbf{C}_{k_0}$ to $\mathbf{C}_{k_1}$ when the object arrives at $\mathbf{f}(n')$, then $m_i^n = 0$ for all n that satisfy $n' \leq n \leq n' + \lfloor \frac{v_{\max}\left(\tau_{\mathtt{proc}}+\tau_i(\hat{\boldsymbol{e}}_i^k, \hat{\boldsymbol{e}}_i^{k+1})\right)}{\Delta l} \rfloor$.

This problem requires the full search on the solution space because g^n changes depending on the combinations of the camera mode values. The solution space consists of all the possible combinations of camera mode values, and its size is $O({N_{\mathtt{cell}}}^{N_{\mathtt{f}}N_{\mathtt{E}}})$. As shown in Section 4.4, $N_{\mathtt{f}}$ exceeds 100 and $N_{\mathtt{E}}$ is more than 20 in our assumed scenarios and resources. Therefore $O({N_{\mathtt{cell}}}^{N_{\mathtt{f}}N_{\mathtt{E}}})$ is still too large to find the optimal solution by a full search practically. As a reasonable solution, our algorithm divides the problem into $N_{\mathtt{cell}} - 1$ independent sub-problems between every pair of adjoining cells, $(\mathbf{C}_k, \mathbf{C}_{k+1})$, by grouping the path fragments

into $N_{\texttt{cell}} - 1$ sections $[A_k, A_{k+1}](k = 1, \cdots, N_{\texttt{cell}} - 1)$, where

$$A_k = \begin{cases} c_1 & (k = 1) \\ \lfloor \frac{c_k + c_{k+1}}{2} \rfloor & (2 \leq k \leq N_{\texttt{cell}} - 1) \\ c_{N_{\texttt{cell}}+1} & (k = N_{\texttt{cell}}) \end{cases}$$

and solve each of them. That is, this division puts a restriction on the camera schedule that all the cameras watch $\mathbf{C}_k$ when the object is at A_k and switch its view from $\mathbf{C}_k$ to $\mathbf{C}_{k+1}$ only once when the object moves from $\mathbf{C}_k$ to $\mathbf{C}_{k+1}$. Here, choosing the values for $A_k(k = 2, 3, \ldots, N_{\texttt{cell}} - 1)$ that maximize Eq. (4.8) itself is an optimization problem that is difficult to solve because of the similar reason described above. Instead we choose the center of each cell for A_k based on the following heuristics: in general, the more cameras see the object, the larger the value of Eq. (4.7) is. As to the number of cameras gazing at a cell, it is reduced in the following two cases; while the object is in $[c_k, A_k - 1]$, some of the cameras switch their view from $\mathbf{C}_{k-1}$ to $\mathbf{C}_k$ in order to follow the object. On the other hand, while the object is in $[A_k, c_{k+1} - 1]$, some of the cameras switch their view from $\mathbf{C}_k$ to $\mathbf{C}_{k+1}$ in order to anticipate the object movement. In both cases, the more frequently those cameras switch their view in the same time, the fewer cameras see the object. Our method reduces such possibilities by making both $[c_k, A_k - 1]$ and $[A_k, c_{k+1} - 1]$ as long as possible. For this reason, we set A_k to the center of each cell, in order to make the value of Eq. (4.8) larger. Each kth sub-problem is formulated as follows:

*k*th sub-problem

Variables $\{n'_{i,k}\}_{i=1,\cdots,N_{\texttt{cam}}} \left(A_k < n'_{i,k} < A_{k+1} - \tau_i(\hat{\boldsymbol{e}}_i^k, \hat{\boldsymbol{e}}_i^{k+1}) \right)$

Objective Function

$$\min_{n \in [A_k, A_{k+1}]} Q(\mathbb{W}(n), \{(\hat{\mathbf{E}}_i(\hat{\boldsymbol{e}}_i^{m_i^n}), \hat{\mathbb{F}}_i(\hat{\boldsymbol{e}}_i^{m_i^n})) | i \in g^n\}) \tag{4.9}$$

$$\text{where } m_i^n = \begin{cases} k & (A_k \leq n \leq n'_{i,k}) \\ 0 & (n'_{i,k} < n \leq n'_{i,k} + \tau_i(\hat{\boldsymbol{e}}_i^k, \hat{\boldsymbol{e}}_i^{k+1}) \\ k+1 & (n'_{i,k} + \tau_i(\hat{\boldsymbol{e}}_i^k, \hat{\boldsymbol{e}}_i^{k+1}) < n \leq A_{k+1}) \end{cases} \tag{4.10}$$

The solutions for these sub-problems can be uniquely mapped to the solution of

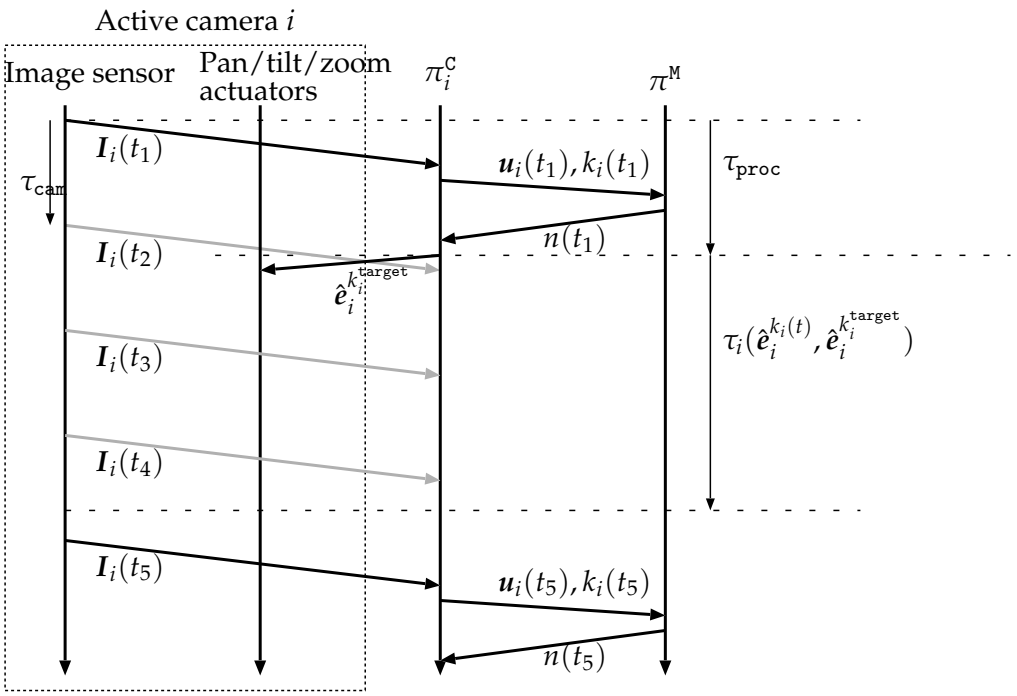


Figure 4.3: Overview of capture process. The camera begins to change its view by the picture corresponding to t_1. Images $\boldsymbol{I}_i(t_2), \boldsymbol{I}_i(t_3), \boldsymbol{I}_i(t_4)$ are discarded because they are taken during the camera motion. ©2010 Springer [YYM10]

the original problem by definition (4.10). The number of solution candidates for each sub-problem is $O((A_k - A_{k-1})^{N_{\mathtt{E}}})$. On the average, $A_k - A_{k-1}$ can be approximated by $\frac{N_{\mathtt{f}}}{N_{\mathtt{cell}}}$. Our algorithm solves this problem using genetic algorithm. A sequence of $n'_{i,k}(i = 1, \cdots, N_{\mathtt{cam}}; k = 1, \ldots, N_{\mathtt{cell}} - 1)$ composes a chromosome, and the fitness function is Eq. (4.9).

4.3.4 Real-time Object Tracking and Camera Control

The cell formation and the scheduling processes compute a pseudo-optimal assignment of the cameras as $\{\hat{\boldsymbol{e}}_i^k\}_{i=1,\cdots,N_{\mathtt{cam}},k=1,\cdots,N_{\mathtt{cell}}}$ and $\{m_i^n\}_{i=1,\cdots,N_{\mathtt{cell}},n=0....,N_{\mathtt{f}}-1}$. Thus the tracking can be performed by measuring the object position $\vec{x}(t)$ and controlling the active cameras in parallel.

Fig. 4.3 summarizes the capture process. It is performed by a computer cluster with $N_{\mathtt{E}}$ camera nodes $\pi_i^{\mathtt{C}}(i = 1, \cdots, N_{\mathtt{cam}})$ and one master node $\pi^{\mathtt{M}}$. These nodes are connected each other to share the object position $\vec{x}(t)$. Every camera node has one active camera connected.

In the following descriptions, we denote the time as t and we assume that all the system clocks on the nodes are synchronized. The measurement of the object 3D position is performed as follows:

The 2D Tracking Process on each $\pi_i^{\mathtt{C}}$

$\pi_i^{\mathtt{C}}(i = 1, \cdots, N_{\mathtt{cam}})$ repeats the following process in every time interval $\tau_{\mathtt{cam}}$.

1. Grab an image $\boldsymbol{I}_i(t)$.
2. If camera i is gazing at one of the cells,
   (a) Store $\{t, \boldsymbol{I}_i(t), k_i(t)\}$. Here, $k_i(t)$ is the cell number that camera i has been gazing at.
   (b) Track the object position on the image and compute its centroid $\vec{u}_i(t)$. The tracking is performed by Condensation algorithm [IB98].
   (c) If $\vec{u}_i(t)$ is successfully computed, transmit $\{t, \vec{u}_i(t), k_i(t)\}$ to $\pi^{\mathtt{M}}$.

The 3D Tracking Process on $\pi^{\mathtt{M}}$

$\pi^{\mathtt{M}}$ repeats the following process in every time interval $\tau_{\mathtt{cam}}$.

1. When two or more sets out of $\{\{t, \vec{u}_i(t), k_i(t)\} | i = 1, \ldots, N_{\mathtt{E}}\}$ have been received, triangulate the 3D position of the object using $\vec{u}_i(t)$ and $\hat{\mathbf{E}}_i(\hat{\boldsymbol{e}}_i^{k_i(t)})$.
2. If the 3D position $\vec{x}(t)$ is successfully calculated, project $\vec{x}(t)$ onto the path and find the corresponding fragment $\mathbf{f}(n(t))$. Transmit $n(t)$ to all $\pi_i^{\mathtt{C}}$.

The camera control is performed by each $\pi_i^{\mathtt{C}}$ as follows:

The Camera Control Process on each $\pi_i^{\mathtt{C}}$

1. Initialization.
   (a) $k_i^{\mathtt{target}} \leftarrow 1$.
   (b) Set the active camera state to $\hat{\boldsymbol{e}}_i^1$.
2. Whenever a new $n(t)$, the fragment number in which the object exists, is received,
   (a) If $k_i^{\mathtt{target}} < N_{\mathtt{cell}}$ and $n(t) \geq n'_{i,k_i^{\mathtt{target}}}$ then:

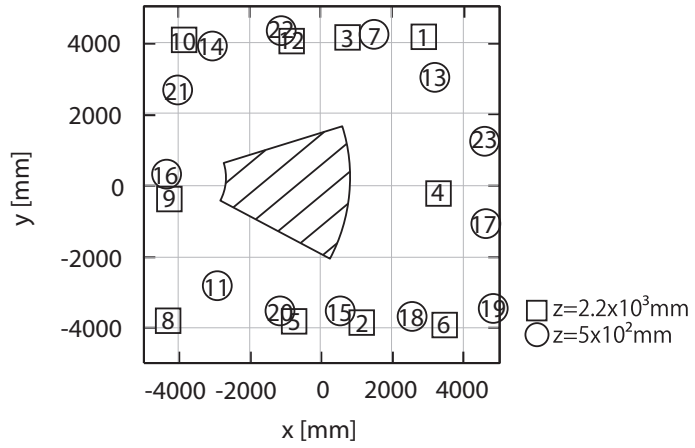


Figure 4.4: The studio and the camera arrangement. The numbers represent the camera positions. ©2009 IEICE [YYMM09]

i. $k_i^{\texttt{target}} \leftarrow k_i^{\texttt{target}} + 1$

ii. Begin switching the active camera state into $\hat{\boldsymbol{e}}_i^{k_i^{\texttt{target}}}$.

4.3.5 3D Video Generation

In the algorithm described above, each $\pi_i^{\texttt{C}}$ stores $\{t, \boldsymbol{I}_i(t), k_i(t)\}$. From these data, a sequence of multi-view images and camera parameters, $\{\boldsymbol{I}_i(t), \hat{\mathbf{E}}_i(\hat{\boldsymbol{e}}_i^{k_i(t)})\}$, which satisfy the four requirements can be obtained. It means that our method can generate a 3D video.

4.4 Experiments and Evaluations

For the experiments described in this section, we arranged 23 active cameras in our 3D video studio, which is about 8 meters square. Each of them is a partially-fixed viewpoint active camera [KWM03] composed of a zoom camera SONY DFW-VL500 and a PTU-46 pan-tilt unit by Directed Perception, Inc. We set up 23 computers as the camera nodes and 1 computer as the master node.

Fig. 4.4 shows the studio and the active camera arrangement. The hatched area is $\hat{\mathbb{F}}_{16}$, the capturable part of the studio by camera 16, when the camera is directed to a person standing at (-1500, 0, 0). As shown by this example, in general, camera views that satisfy Req. 3 are limited to part of the studio space and cannot cover all the studio space at one time.

As the resources required by the algorithm, $\tau_i(e_i, e'_i)$ was given based on measured dynamic characteristics of each active camera. $K_i(\mathbf{V})$ was made up to find the camera control value that captures $\mathbf{V}$ near the center of image. As to $\texttt{visible}(\mathbf{V}, \vec{p}_j, (\hat{\mathbf{E}}_i, \hat{\mathbb{F}}_i))$, a function to compute self-occlusion of the object shape was given because there is no other object that occludes the object in our studio.

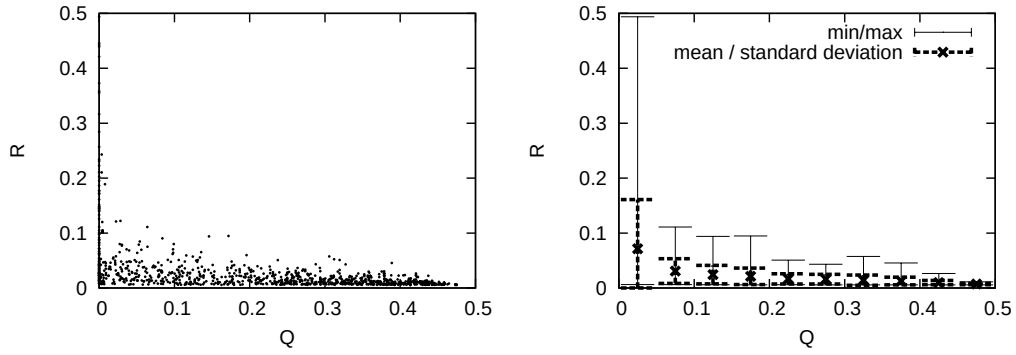
The following subsections describe the details of three experiments conducted under this environment.

4.4.1 Evaluation of the Visual Coverage Function

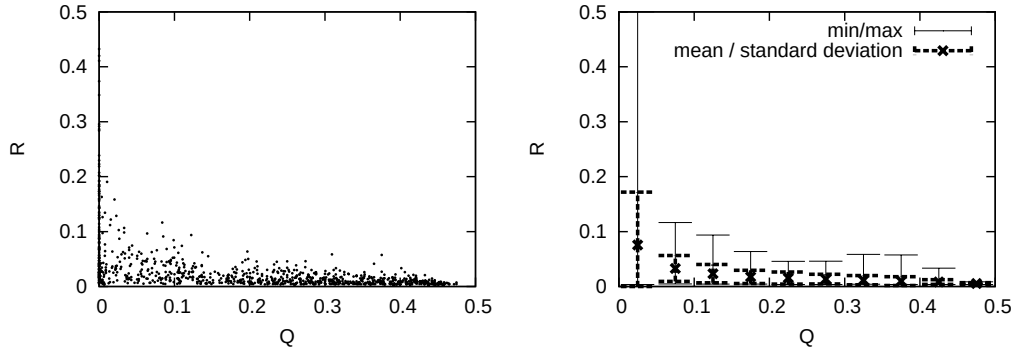
First, we show that the objective function defined by Eq. (4.8) is effective for estimating the visual coverage of real objects. For this purpose, we compared the evaluation function value Q, which is defined by (3.7), to the non-observed surface rate R for several different scenes. Here R is defined as the ratio of invisible patches from any camera, to the entire object surface. The lower R is, the better Req. 2 is satisfied.

We used 2 types of digitized human shape model as test data. We generated 1000 virtual camera configurations for each using the path and camera configuration shown in Fig. 4.6(a). Each of them was generated by putting an object at a random position on the path, randomly choosing a subset of cameras to be used, and then setting up their control values by K_i. For the bounding volume $\tilde{\mathbb{V}}(\vec{x})$, which is used for computing the evaluation function Q, a sphere centered at $\vec{x}$ was used. The reason why we chose such $\tilde{\mathbb{V}}$ is described in Section 4.4.2.

Fig. 4.5 shows the result distribution of (Q, R). Each black dot represents one camera configuration. Because the visual coverage is largely affected by the shape of target object and relative positions of the cameras to the object, there is a large variance in R for any Q. Some parts of the object such as the soles of the feet are physically unobservable and thus R never reaches zero. The maximum value of R for each Q, however, tends to be the lower for the higher Q. It shows that Eq. (3.7) can estimate the worst-case visual coverage for objects that have unknown shape, using the surrogate shape model $\tilde{\mathbb{V}}$. The results proves that optimizing camera views for Eq. (3.7) leads to better capture of multi-view video from the standpoint of Req. 2.



(a) Shape 1



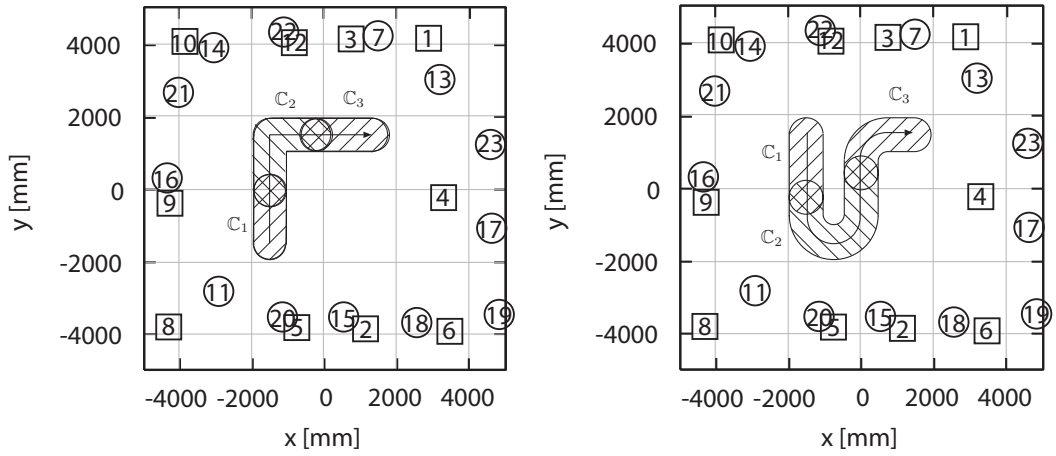
(b) Shape 2

Figure 4.5: Distribution of (Q, R) — the visibility function and non-observed surface ratio — for two human shape model data. The left column shows the distribution of samples. The right column shows the mean, standard deviation, minimum, and maximum values of R in every 0.05 interval of Q. ©2009 IEICE [YYMM09]

4.4.2 Tracking Experiments

As an experiment to show the effectiveness of our method, we captured 3D videos of a walking person in a widespread area. The value of Eq. (3.7) changes depending on the relative positions of the cameras to the object. Consequently, the cell formation and camera control scheduling in our algorithm is also affected by them. Hence, we prepared two scenarios with different paths. The arrows in Figs. 4.6(a) and 4.6(b) stand for the given path $\mathbb{L}$ for each scenario. We assumed that the height of the target person is 1.8 meters, and that he moves along the given path with varying speed slower than 0.5 meters per second. The resolution requirement was set to $s = 8$[mm/pixel].

First, we attempted to capture the whole body of the target. Hence, $\tilde{\mathbb{V}}(\vec{x})$ was specified as double-stacked spheres at $\vec{x}$ whose radii are 0.45 meters. With these



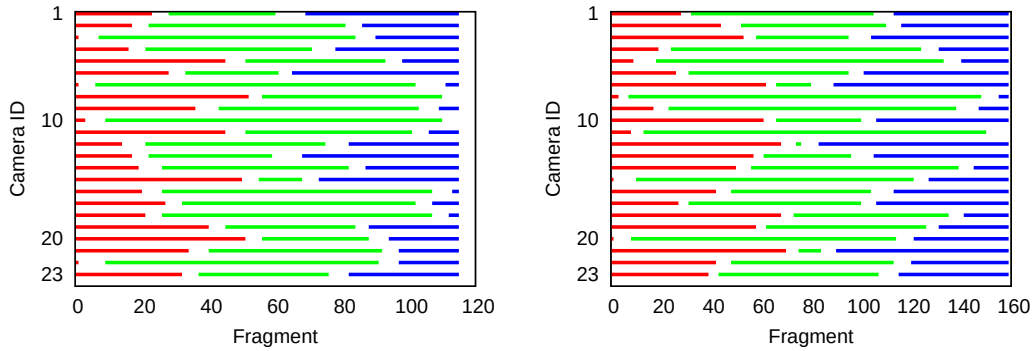
(a) Scenario 1 (b) Scenario 2

Figure 4.6: Path and cells. The arrows represent the given path for each scenario. The hatched parts represent common view volumes of the cells. ©2009 IEICE [YYMM09]

scenarios and the resources, our cell formation algorithm detected that it was impossible to capture the target. This is due to our studio setup. Some cameras were too close to the paths. For example, when the object is at (-1500, 0, 0), camera 16 cannot capture all part of the target because the projected height on its image exceeds the image height. This result shows one of the benefits with our method; it can judge if the requirements can be satisfied before capture.

Second, we attempted to guarantee that the upper half of the body is successfully captured and verified the generated 3D video of it. $\tilde{\mathbb{V}}(\vec{x})$ was specified by a single 0.45 m-radius sphere 0.9 m above the floor, which includes the upper half part of a standing person. Other resources and scenarios were the same as the first experiment.

We ran the cell formation algorithm with these inputs. The cell formation algorithm was implemented on a computer with Xeon 3.6 GHz CPU. The cells were successfully generated and the running time was less than 5 seconds in both cases. The hatched areas in Fig. 4.6 visualize the generated cells for each scenario. Then, the camera parameters for each cell were calibrated; the intrinsic parameters were estimated by Zhang's method [Zha00], the extrinsic parameters were estimated by the eight-point algorithm [HZ04] for each pair of cameras with 2D-to-2D correspondences of unknown 3D points, and then refined through a bundle adjustment process which minimizes the sum of symmetric epipolar distances of all cameras. After that, camera control was scheduled based on the generated cells.



(a) Scenario 1 (b) Scenario 2

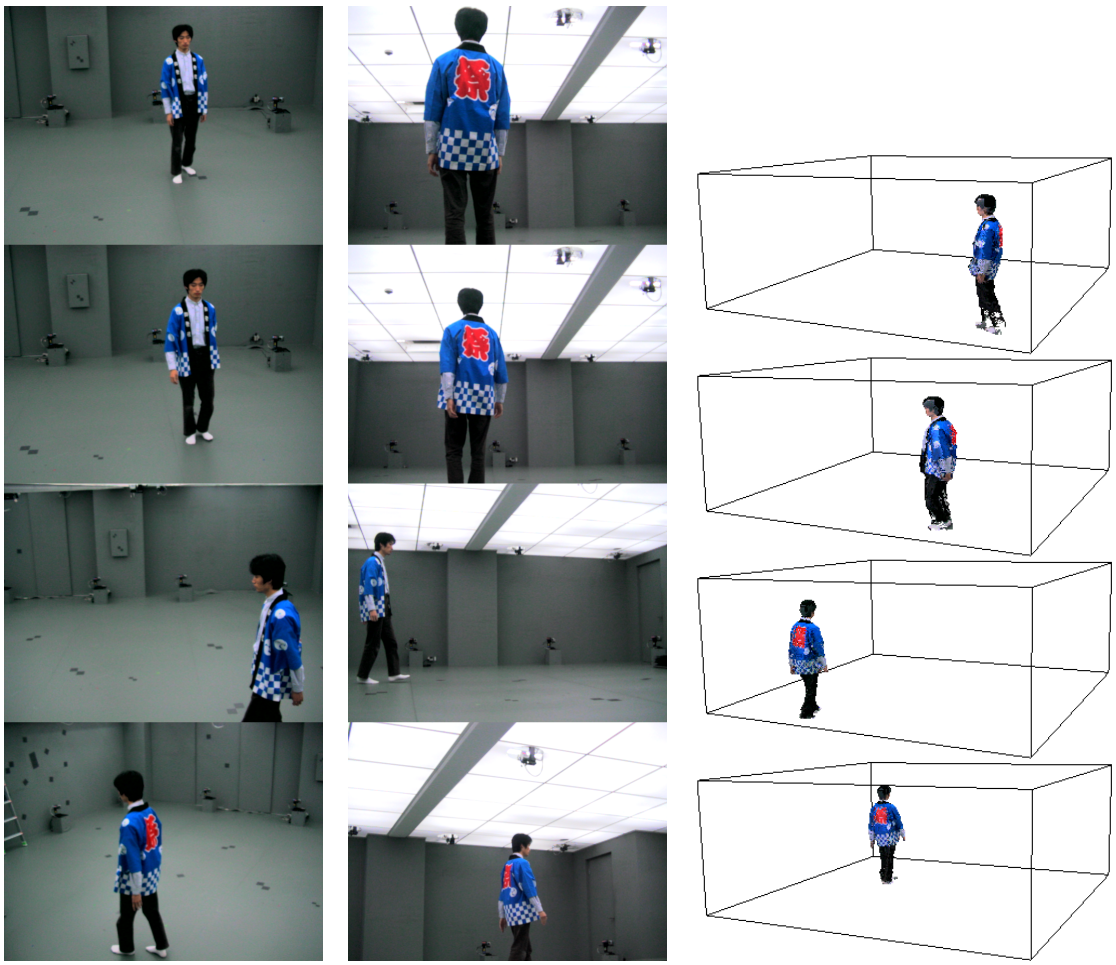
Figure 4.7: Optimized schedules. Vertical axis represents each active camera. Horizontal axis represents target position by fragment number. Each solid line expresses an interval where the camera watches a cell, and blank parts represents the intervals where the camera switches its view to the next cell. ©2009 IEICE [YYMM09]

Table 4.5: GA optimization details

Number of units per generation	2000
Crossover method	Uniform crossover
Mutation	Replace one gene, $n'_{i,k}$, with a random value per unit. No mutation for top 200 units.
Generations	2000

The camera control scheduling algorithm was implemented on a computer with Xeon 3.6GHz CPU. The parameters for the genetic algorithm in the optimization process is described in Table 4.5. The computation times were about 80 hours. Fig. 4.7 visualizes the result schedules. In these figures, vertical axis represents each active camera and horizontal axis represents target position by fragment number n. Each solid line expresses an interval where the camera is gazing at a cell, and blank parts represents the intervals where the camera is switching view to the next cell. We can see that these schedules are avoiding that too many cameras are switching view to the next cell simultaneously in both cases.

A walking person was tracked for a multi-view video by the cell-based tracking algorithm using these cells and schedules. Then 3D video was generated from the tracking records. Fig. 4.8 shows the captured multi-view video and the generated 3D video. These results indicate that the upper half of the body was successfully captured and 3D video of that part could be generated. Thus it was



(a) Camera #1 (b) Camera #6 (c) Generated 3D video

Figure 4.8: Captured multi-view video and generated 3D video, at frame 15, 54, 104, and 151 ©2009 IEICE [YYMM09]

shown that our method can produce high-resolution 3D video of the specified object.

On the contrary, the lower half was not successfully captured in some frames. This was due to mechanical limitations of the active cameras. This resulted in the lack of texture on the legs, as shown in Fig. 4.8(c).

In summary, our method has realized these two functions. First, it has realized 3D video production of an object moving in a widespread area. Second, our algorithm can detect that the four requirements cannot be satisfied before tracking.

4.4.3 Performance Evaluation

For the quantitative evaluation of our method, we compared it with a pair of possible methods with static cameras, from a viewpoint of the effectiveness of the camera usage. We define the following two indices.

$$\text{(Viewpoint usage at frame } z) = \frac{|G(z)|}{N_{\mathtt{cam}}} \tag{4.11}$$

$$\text{(Pixel usage at frame } z) = \frac{1}{|G(z)|} \sum_{i \in G(z)} \frac{N_i^{\mathtt{P}}(z)}{N_i^{\mathtt{I}}} \tag{4.12}$$

$G(z)$	$\{i \mid$camera i not in motion but captured the object$\}$
$N_i^{\mathtt{P}}(z)$	Number of pixels occupied by the object in the image of camera i
$N_i^{\mathtt{I}}$	Number of pixels in image i. e.g. 307200(=640$\times$480) for VGA.

The larger these indices are, the more information can be obtained for 3D video generation from images. Thus, it leads to the high fidelity of 3D video. As mentioned in Section 4.1, there is a trade-off between the two indices when methods with static cameras are used, especially when capturing an object moving in a widespread area. As mentioned in section 4.1, we can think of two methods using static cameras: (1) the "view-optimized" method, which gives higher priority to the viewpoint usage and (2) the "resolution-optimized" method, which gives higher priority to the pixel usage.

We used the same scenario as the experiment in Section 4.4.2. In the fixed camera settings for the view-optimized method, views of every camera were adjusted in order to include the entire volume where the object passes. Hence, lenses with very wide angle of view were virtually generated. For the resolution-optimized method, the cameras were divided into 3 groups and assigned to watch one of the three cells generated by the cell formation algorithm. There are 3^{23} combinations to assign 23 cameras to one of the 3 cells independently. Thus we randomly generated 10^{12} combinations and chose the best one that maximizes Eq. (3.7).

The two indices are also sensitive to the shape of the object, as well as the camera configurations. In order to evaluate different camera control methods using the same dynamic scene, we first generated a 3D video using our method and then simulated the other two methods by synthesizing virtual images from the 3D video data.

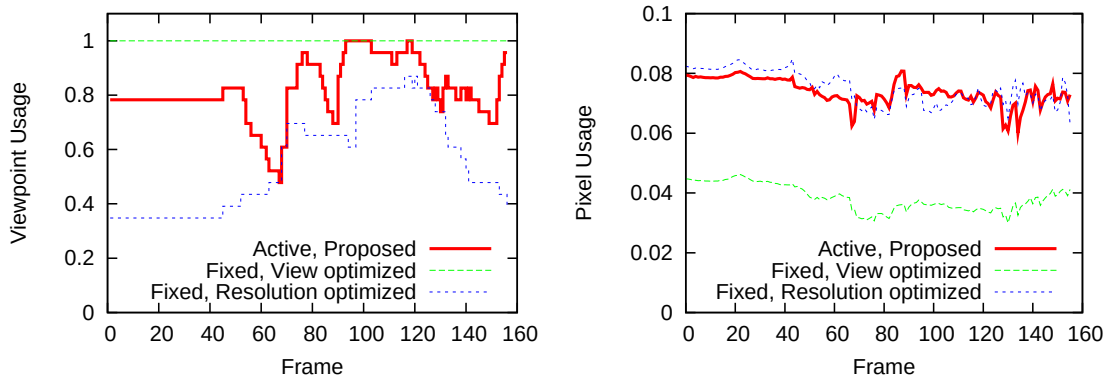


Figure 4.9: Viewpoint and pixel usage in scenario 1 ©2009 IEICE [YYMM09]

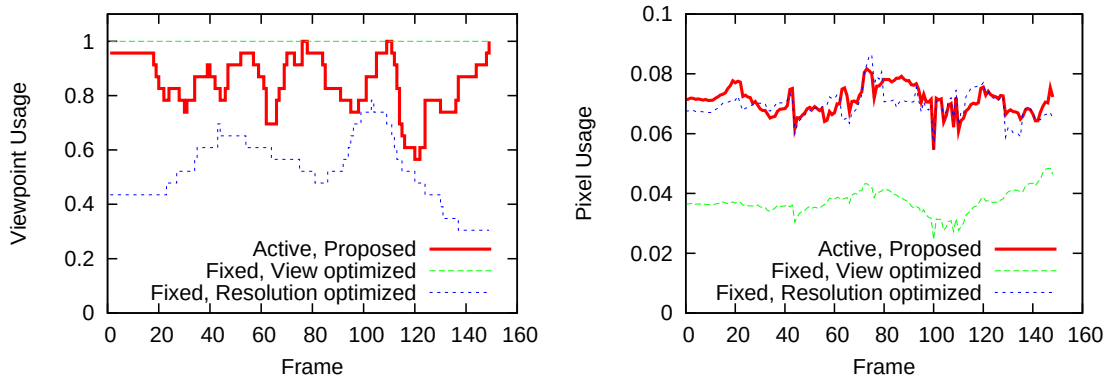


Figure 4.10: Viewpoint and pixel usage in scenario 2 ©2009 IEICE [YYMM09]

Figs. 4.9 and 4.10 shows the two indices for each method. These results show the trade-off problem between viewpoint usage and pixel usage in the methods using static cameras. For example, as shown in Fig. 4.9, if higher priority is given to viewpoint usage, it is high through the sequences but the pixel usage is lower, since the object is projected small into those camera images. The same trade-off is also shown in Fig. 4.10. As shown, the methods using static cameras cannot capture without losing one of them. On the contrary, our method can improve viewpoint usage while keeping the same pixel usage with the resolution-optimized method. As to the viewpoint usage, though it changes largely depending on the object position and the path, our method has improved the worst values in the sequences compared to the resolution-optimized method in both case.

Finally, we discuss the minimal number of static cameras that would be necessary to achieve the same quality of 3D video with our methods, with these scenarios. Resolution-optimized method retained the same pixel usage with our method. Therefore, the average viewpoint usage gives a rough estimation of the

3D video quality. The average viewpoint usage is 83% with our method whereas it is 54% with the resolution-optimized method. From these figures, we estimate that our method can attain the viewpoint usage about 1.6 times higher than the resolution-optimized method, while keeping the same pixel usage. Consequently, we estimate that roughly 36 fixed cameras are required.

To summarize, our method is effective from the standpoint of viewpoint usage and pixel usage as well.

4.5 Summary

We formulated the active camera control problem for 3D video capture, when a target object moves along a given path. We proposed a cell-based capture algorithm as a practical solution and showed that it can capture high-resolution 3D video improving camera and pixel usage.

Chapter 5

Cell-Based 3D Video Capture of a Freely Moving Object

The algorithm described in Chapter 4 cannot be applied when the object's trajectory is not given in advance. In this chapter, we tackle the 3D video capture problem of an object that moves freely on a flat floor. The fundamental problem is how to ensure continuous 3D video capture of the object for its arbitrary direction of movement. The main idea of the algorithm is the cell arrangement based on the regular hexagon tessellation. Under the condition that the upper limit of the time required to change state of active cameras is given by a constant value in advance, the size of the cells can be uniquely determined based on it and the maximum speed of the object. Assignment of the active cameras to the cells based on the 3-coloring of the cells can ensure continuous object capture with at least one third of the active cameras.

The effectiveness of the method is shown by experiments. Simulation result shows the quantitative improvement of the 3D video. An experimental capture system is built to show the effectiveness of the method in the real world.

5.1 Problem Description

The scope of this chapter is summarized as follows:

- Only one target object.
- The object movement is restricted to a given target area.
- The object's maximum velocity is given.

- The object's size is given.
- Active cameras are set up to surround the target area so that the object there can be observed from different directions.
- Upper limit of the time required for changing active camera state is given by a constant value.
- The object is not occluded by other objects from any camera.
- The resolution requirement is specified by the lowest allowable resolution.

5.2 Approach

As declared in Chapter 3, there are two essential design decisions to be made when building a cell-based algorithm; how to divide the space into cells and how to control active cameras based on the cells. Since the object can move freely, shape reconstruction should be performed equally well regardless of the object position and its direction of movement. Thus the division and the cell shape should be both homogeneous and isotropic. As a natural solution, we adopt a regular hexagon tessellation for cell arrangement.

The second design decision is the camera control rule based on these cells to ensure the continuous 3D video capture of a freely-moving object. Our cell-based method reduces this problem to an assignment problem of the cameras to the cells. The hexagonal cell arrangement has a 6-neighborhood structure. Thus any object movement can be expressed by a movement to one of the six cells. The capture system must be ready to capture the object for all of these directions. One intuitive and possible solution is to assign the cameras to the cell where the object exists and the six cells next to it. However, it is not desirable from the standpoint of Req. 2, since it can only ensure capture by one seventh of the cameras. Thus we have devised a more efficient way by reducing the number of the cells to be covered. For example, another simple and intuitive method is to divide the cameras into two groups and to assign them to watch the cell where the object exists and the next cell that the object is moving to. However, this method fails when the object passes across a cell vertex where three cells are in contact with each other. For example, when the object is located at $\vec{x}$, as shown in Fig. 5.1, it can either go into cell B, C, or stay in cell A. The capture system must anticipate

all of these three cases by watching the three cells. This proves that the capture system must be able to observe at least 3 cells simultaneously.

The algorithm described in this chapter captures the three nearest cells from the object using three homogeneous camera groups. We prove that this is also a sufficient number. For example, assume that the object approaches cell D that is currently the fourth-nearest. D becomes the third-nearest cell instead of C when the object arrives at $\vec{x'}$. At this point, the capture system no longer needs to watch cell C. Consequently, the cameras that have watched cell C can change views to watch cell D. Additionally, the distance to D is not zero but equal to or longer than $R/2$, where R is the radius of the cells as described in Figure 5.1. Assuming that the object is not "too fast" (discussed in section 5.4.2 more precisely), the cameras can finish their motions before the object reaches cell D. Also note that if the object goes back to cell C crossing over the dashed line in Fig. 5.1 again, the cameras can resume capturing cell C before the object gets into cell C. Similarly, when the object approaches cell G, the cell can be covered by the same cameras. Cells E and F can be covered by the cameras currently watching cell B. This discussion is valid regardless of the cell the object lies in, by associating the three camera groups to three sparse subsets of the cells as shown in Fig. 5.2. In this way, our method ensures continuous capture of the object at least with one third of the cameras. To generalize, the maximum number of cells that share a single vertex gives the minimum number of required camera groups. Hence a hexagonal cell shape is also the best from this standpoint.

5.3 Formulation

Active Camera Model

We assume that each active camera can be approximated by a partially-fixed viewpoint pan-tilt-zoom (PFV-PTZ) camera model[KWM03] [Mat98]. A PFV-PTZ camera is a camera whose projection center is encased in a limited volume around the rotation center. The changes in their projection centers are relatively small when capturing objects far enough from such cameras. We approximate the projection center of an active camera by a steady point regardless of the pan, tilt, zoom and focus motions of the camera. Therefore the camera positioning problem can be separated from the active camera control problem. We also assume that the cameras are mounted on stations surrounding the target area. The cam-

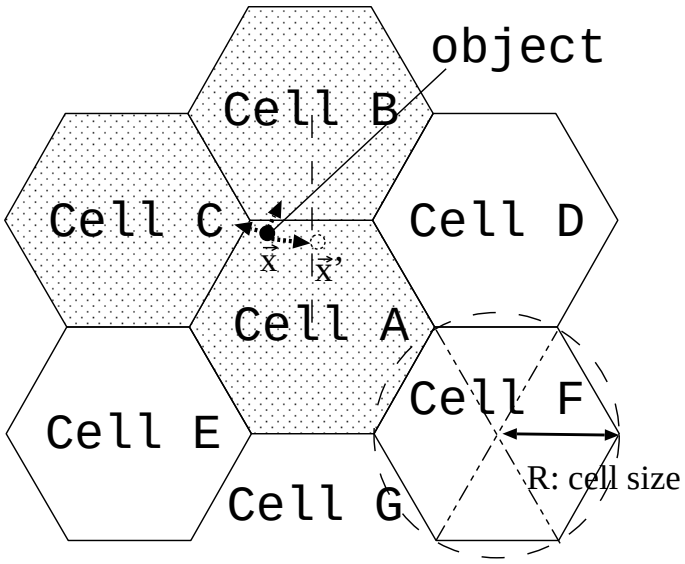


Figure 5.1: Cells and the object. A is the cell where the object is present, and thus the nearest cell from the object. B, C and D are the second-, third-, and fourth-nearest cells, respectively. The capture system must be ready to capture A, B and C, especially when the object is nearby the vertex shared by these cells. On the other hand, cell D can be covered by the cameras that watched C before the object approaches D. ©2010 IPSJ [YYNM10]

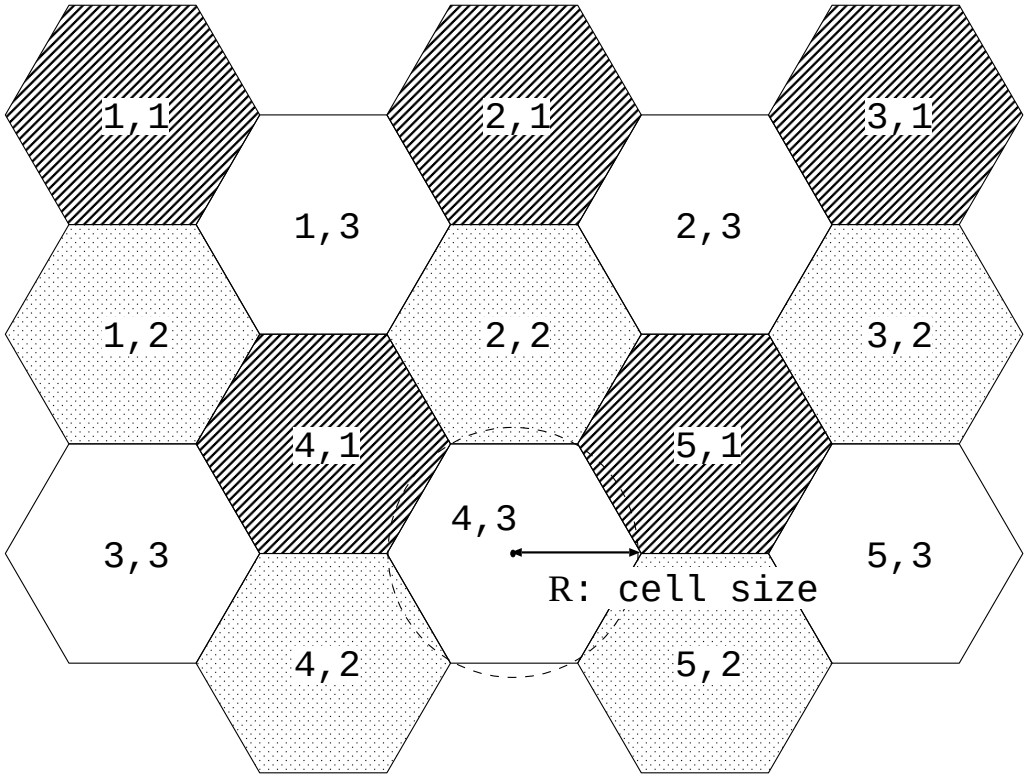


Figure 5.2: Cell assignment for three camera groups. The numbers in the figure represents k, l = cell index and group index. ©2010 IPSJ [YYNM10]

eras change directions and focal lengths according to each camera control parameter, which consists of pan, tilt, zoom and focus positions. We denote camera control parameters for camera i as $\mathbf{e}_i$. Since these parameter changes accompany physical motions of the cameras, there exist time lags between the transmission of these parameters and the end of the corresponding motions. The lengths of these time lags depend on the current and target state of each camera, but are guaranteed to be shorter than τ_s seconds.

Target Object

The target object moves within a target area at varying speeds that is slower than $v_{\mathtt{max}}$. The object's size is specified by the user as a bounding volume. The object can change its shape within a bounding volume located at the same position with the object, as shown in Fig. 5.3. Any shape can be used as the bounding volume, though it would be natural to use a solid of revolution, e.g. a cylinder or a hemisphere, since the object is supposed to move in arbitrary directions.

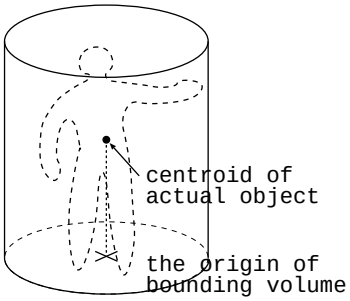


Figure 5.3: An example of bounding volume that represents the maximum allowable size of the object, in case of a cylinder. ©2010 IPSJ [YYNM10]

Table 5.1: Scenario

$\mathbf{D} \subset \mathbb{R}^2$	Target area, within which the object is allowed move during capture.
$v_{\mathtt{max}}$	Maximum speed of the object.
$\mathbf{B} \subset \mathbb{R}^3$	Bounding volume of the object.
s[mm/pixel]	Lowest allowable spatial resolution.

We adopt the world coordinate system that has the origin at the center of the target area and the z axis directed upright. We represent the object state by the projection point of its centroid to the floor and denote it by the 2D coordinates $\vec{x} = (x, y)$. In other words, we define a 2D state space which can be mapped to the floor plane and represent the object state by a point in the state space.

By making the assumptions about the active cameras and the object, one problem instance for our algorithm consists of the variables listed in Tables 5.1 and 5.2. A scenario is given by users reflecting the scene to be captured. On the other hand, resources are governed by the studio equipment. The output of our algorithm is a 3D video of a moving object in a wide-area. The algorithm consists of the following processes.

1. Camera grouping
2. Cell generation
3. Camera view adjustment
4. Camera calibration
5. Real-time tracking
6. 3D Video generation

Table 5.2: Resources

$N_{\mathtt{cam}}$	Number of active cameras.
$\mathrm{O}_1, \mathrm{O}_2, \ldots, \mathrm{O}_{\mathrm{N}_{\mathtt{cam}}}$	Active camera positions.
$\tau_{\mathtt{s}}$	Upper limit of the time required for changing active camera state.
$\tau_{\mathtt{cam}}$	Video capture interval.

5.4 Algorithm

5.4.1 Camera Grouping

Due to the reasons outlined in section 5.2, the first step of our algorithm divides the cameras into three groups. Since only one of the groups is available for capture at the worst case, every group should have enough cameras to satisfy Req. 2 by itself. Accordingly, it is desirable that the cameras in each group are equally distributed in all the directions. Thus we adopt the following scheme.

1. Represent the camera positions by spherical coordinates (r_i, θ_i, ϕ_i) where θ_i is the zenith angle that satisfies $0 \leq \theta_i \leq \pi/2$, and ϕ_i is the azimuth angle that satisfies $0 \leq \phi_i < 2\pi$.

2. Sort the camera indexes according to ϕ_i. Let the sorted camera indexes be $i_1, i_2, \ldots, i_{N_{\mathtt{cam}}}$.

3. For all $n = 1, 2, \ldots, N_{\mathtt{cam}}$, assign camera i_n to group $(n - 3\lfloor n/3 \rfloor + 1)$.

In the following descriptions, we express the camera groups by $\mathbf{A}_1, \mathbf{A}_2$ and $\mathbf{A}_3$ where $\mathbf{A}_l = \{i | \text{camera } i \text{ is assigned to group } l\}$. We also use the notation $l(i)$ to mean "the group index that camera i is assigned to".

5.4.2 Cell Generation

The second step divides the target area into cells using regular hexagon tessellation shown in Fig. 5.2 and assigns them to the camera groups. Our algorithm represents a cell as a joint, close subset of the target area. The cells are assigned to the three camera groups exclusively as shown in Fig. 5.2. Thus we denote cells using two indexes; group index l and cell index k.

$$\mathbf{C}_{k,l} \subset \mathbf{D}(l = 1, 2, 3; k = 1, 2, \ldots, N_l^{\mathtt{cell}}) \tag{5.1}$$

Here, $N_l^{\mathtt{cell}} \in \mathbb{N}$ is the number of the cells assigned to group l. We also define the following symbols. They are precomputed later in the following steps.

$\hat{\boldsymbol{e}}_i^k$ Camera control parameters for directing camera i to $\mathbf{C}_{k,l(i)}$. Consists of pan, tilt and zoom values.

E_i^k Camera parameters — intrinsic and extrinsic, geometric and photometric parameters — of camera i with the state specified by $\hat{\boldsymbol{e}}_i^k$.

We define the distance between the object located at $\vec{x}$ and a cell $\mathbf{C}_{k,l}$ by (5.2). It is defined as the distance from the object to the nearest point in the cell.

$$d(\vec{x}, \mathbf{C}_{k,l}) = \min_{\vec{q} \in \mathbf{C}_{k,l}} ||\vec{q} - \vec{x}|| \tag{5.2}$$

We use regular hexagon tessellation shown in Fig. 5.2 for the shape and the arrangement of cells by the reasons described in section 5.2. Cell arrangement by the regular hexagon tessellation has four degrees of freedom: 2D displacement, rotation, and the size of the cells. The displacement and the rotation does not affect the 3D video capture as long as the cameras are not too close to the target area. This is because the cameras in each group are equally distributed in all directions by the camera grouping step described in section 5.4.1. Thus we give these parameters manually. On the other hand, the size of the cells is critical and must be designed while considering the camera control rule.

Camera Control Rule and Cell Size Our camera control method directs all the cameras in $\mathbf{A}_{l_0}$ to the nearest cell out of $\{\mathbf{C}_{k,l} | l = l_0\}$ from the object. If there are two or more nearest cells, the cameras should be directed to one of those cells. This control rule can be represented by functions that map an object position to a cell index:

$$f_l(\vec{x}) = \operatorname*{argmin}_k d(\vec{x}, \mathbf{C}_{k,l}) \tag{5.3}$$

Our cell arrangement shown in Fig. 5.2 gives the camera control rule as shown in Fig. 5.4. The cameras begin to change views when the value of (5.3) changes as the object moves. In other words, the cameras in group l begin to switch views when the object goes across a line where the distance to the nearest two cells are the same, as shown in Fig. 5.4. We call such lines "rule border" for group l.

Switching camera views from one cell to another cell requires a certain amount of time not longer than $\tau_{\mathtt{s}}$. During these periods, the images from such "in-motion" cameras cannot be used for shape reconstruction because they are not

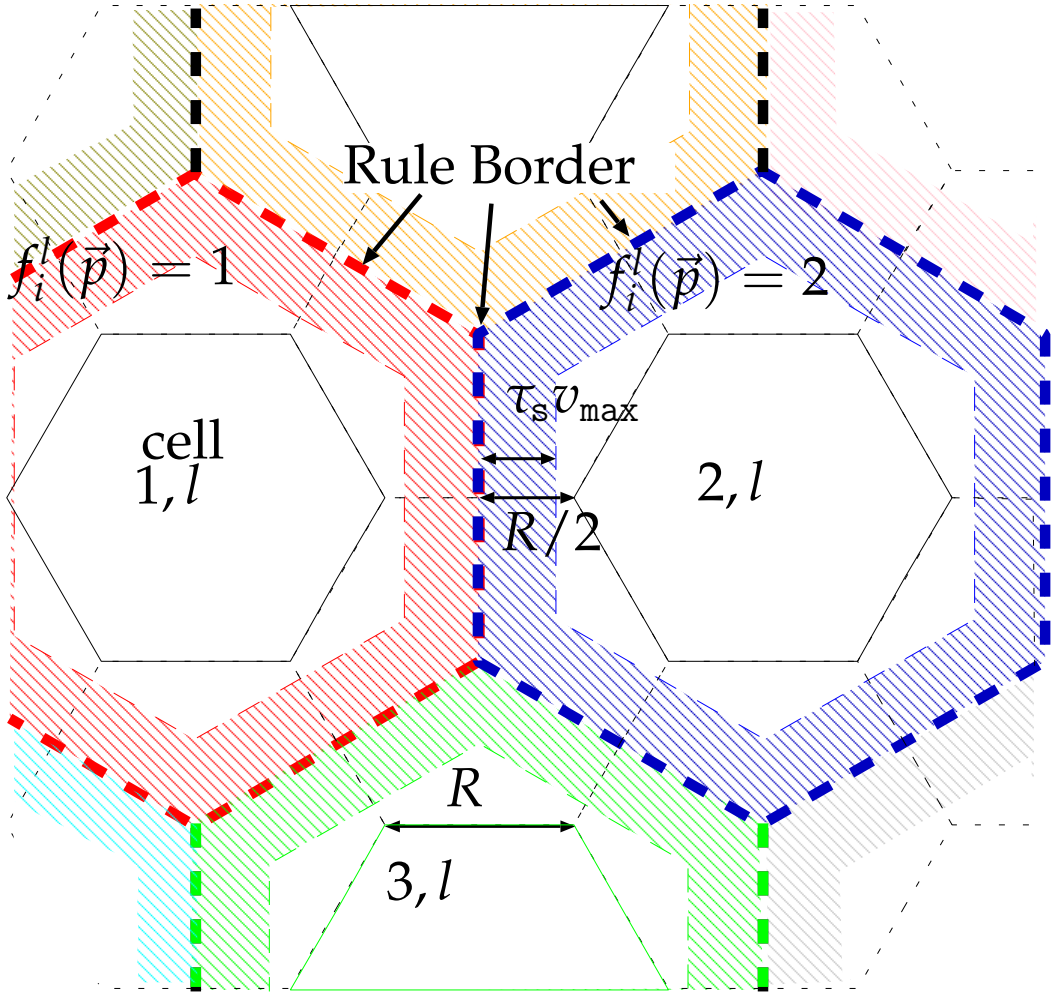


Figure 5.4: Visualization of camera control rule for a single group l. Cells are represented by solid line. Dashed thick lines are "rule borders." The cameras in this group begin their motions to change views when the object crosses over a rule border. The hatched regions represents the area where the distance from any of the rule borders is greater than $\tau_s v_{max}$. It is guaranteed that if the object is inside the hatched area, the cameras in the group has finished their motions and are capturing cell $f_l(\vec{x})$. ©2010 IPSJ [YYNM10]

calibrated. Meanwhile, the object can move by $\tau_{\mathtt{s}} v_{\mathtt{max}}$ in the worst case. It means that where the distance from the nearest rule border for group l is shorter than $\tau_{\mathtt{s}} v_{\mathtt{max}}$, the object is not guaranteed to be captured by the cameras in $\mathbf{A}_l$. Thus the necessary and sufficient condition for cameras in $\mathbf{A}_l$ to ensure observation of the object in cell $\mathbf{C}_{k,l} (k = 1, 2, \ldots, N_l^{\mathtt{cell}})$ with respective to state $\hat{\boldsymbol{e}}_i^k$, for arbitrary object movement, is represented by Eq. (5.4). Since the object exists in one of the cells at any one time, this is also a sufficient condition to capture the object continuously, by at least one of the three camera groups.

$$R \geq 2\tau_{\mathtt{s}} v_{\mathtt{max}} \tag{5.4}$$

Eq. (5.4) means that the minimum size of the cells is limited by $\tau_{\mathtt{s}}$ and $v_{\mathtt{max}}$. The larger the cell is, the larger space the camera views must cover and thus the spatial resolution becomes lower. In order to maximize the spatial resolution, we adopt the minimum allowable cell size as shown by Eq. (5.5).

$$R = 2\tau_{\mathtt{s}} v_{\mathtt{max}} \tag{5.5}$$

To summarize, our cell generation algorithm is described as follows.

1. Compute the cell size by Eq. (5.5).
2. Give a cell position and rotation manually.
3. Generate other cells by dividing $\mathbf{D}$ according to the regular hexagon tessellation pattern shown in Fig. 5.2.

5.4.3 Camera view adjustment

This step adjusts dedicated camera control parameters in order to watch each cell. In order to guarantee the object capture at any point in a cell, every camera view in group l should include the Minkowski sum of $\mathbf{C}_{k,l}$ and $\mathbf{B}$, as shown in Fig. 5.5. We call this volume "the common view" of cell k, l and denote it by $M_{k,l}$. Thereby the condition is described by Eq. (5.6).

$$M_{k,l} \subset \hat{\mathbb{F}}_i(\hat{\boldsymbol{e}}_i^k) \tag{5.6}$$

If $\hat{\boldsymbol{e}}_i^k$ does not exist, it means that the object cannot be captured with our method for the given scenario. Our algorithm terminates at this step in this case.

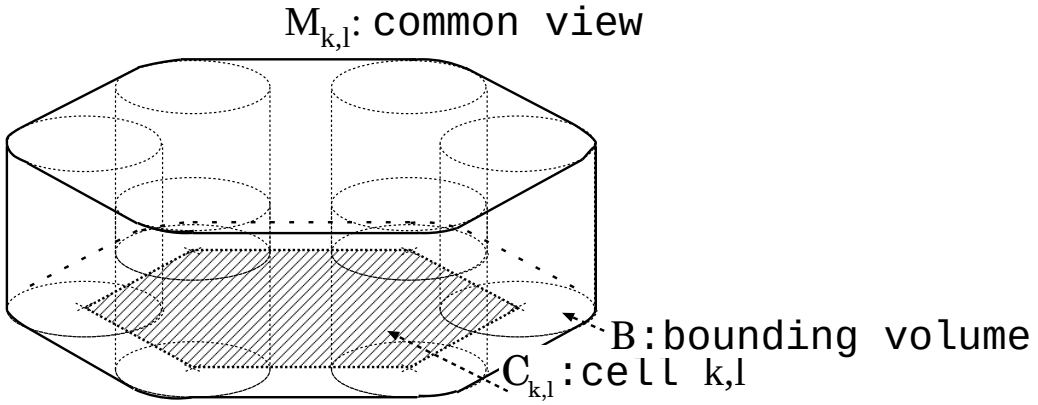


Figure 5.5: Minkowski sum of a cell and the bounding volume. ©2010 IPSJ [YYNM10]

The algorithm to find $\hat{\boldsymbol{e}}_i^k$ should be designed depending on the structure of the active cameras in use. We describe one example algorithm that gives $\hat{\boldsymbol{e}}_i^k$ for a PFV-PTZ active camera: First, adjust the zoom value so that the spatial resolution becomes s at the farthest point in $M_{k,l}$. Next, adjust the focus value so that the nearest and the farthest points in $M_{k,l}$ to the camera is included within the field of view. Finally, adjust the pan/tilt angles so that $M_{k,l}$ is included in the image frame of the camera. If one or more of these parameters do not exist, then there is no solution.

5.4.4 Camera Calibration

We calibrate the active cameras and obtain E_i^k for all $i = 1, \ldots, N_{\mathtt{cam}}, k = 1, \ldots, N_{l(i)}^{\mathtt{cell}}$. At this step, we do not utilize any explicit active camera model such as the PFV-PTZ camera model. All the active cameras can be regarded as fixed cameras, while they watch one of the cells. Namely, we equivalently have $\sum_{l=1}^{3} ||\mathbf{A}_l|| N_l^{\mathtt{cell}}$ "virtual fixed cameras" where $||\mathbf{A}_l||$ is the number of the active cameras forming camera group l. Any existing camera calibration methods for fixed cameras, such as Zhang's[Zha00] and Svoboda's[SMP05], can be applied to these virtual fixed cameras. Since this is an off-line process separated from the following tracking process, we can use any calibration targets such as calibration charts or a point light source to obtain camera parameters accurately and robustly.

5.4.5 Real-time tracking

The cell generation and the camera view adjustment process define the camera control rule — how all the cameras should be controlled for any object position. Additionally, the camera calibration process gives accurate camera parameters that enable 3D positioning of the object by the images. Thus the tracking can be performed by measuring the object position from the images and then controlling the active cameras in parallel.

We describe our tracking process using a model of networked computers. The capture process is performed by $N_{\mathtt{cam}}$ camera nodes $\pi_i (i = 1, 2, \ldots, N_{\mathtt{cam}})$, which have one camera connected to each, and one master node $\pi_{\mathtt{M}}$. The nodes are connected to each other to share the object position. In the following descriptions, we denote the time by t and we assume that all the system clocks on the nodes are synchronized.

The measurement of the object 3D position is performed as follows:

2D Tracking Process on each π_i

Repeat the following procedure in every time interval $\tau_{\mathtt{cam}}$.

1. Grab an image $I_i(t)$.
2. If the camera is not in motion but watching one of the cells in group $l(i)$, let its cell index be $k_i(t)$
   (a) Store $(t, I_i(t), k_i(t))$.
   (b) Find the object in the image and compute its centroid coordinate $\vec{u}_i(t)$.
   (c) If $\vec{u}_i(t)$ is successfully computed, transmit $(t, \vec{u}_i(t), k_i(t))$ to $\pi_{\mathtt{M}}$.

3D Tracking Process on $\pi_{\mathtt{M}}$

1. When 2 or more sets out of $\{(t, \vec{u}_i(t), k_i(t)) \,|i = 1, \ldots, N_{\mathtt{cam}}\}$ have been received,
   (a) Compute the 3D position of the object, $\vec{\hat{x}}(t) = (x, y, z)$, by triangulation $\vec{u}_i(t)$ and $E_i^{k_i(t)}$.
   (b) Transmit $\vec{x}(t) = (x, y)$ to all π_i.

On the other hand, the camera control is performed as follows:

Camera Control Process on each π_i

1. Whenever a new $\vec{x}(t) = (x, y)$ is received,
   (a) If $f_{l(i)}(\vec{x}(t)) \neq f_{l(i)}(\vec{x}(t - \tau_{\mathtt{cam}}))$:
      i. Transmit $\hat{e}_i^{f_{l(i)}(\vec{x}(t))}$ to the active camera and begin to switch its posture, zoom and focus.

5.4.6 3D Video generation

In the algorithm described above, each π_i stores $(t, I_i(t), k_i(t))$. From this data, a sequence of multi-view images and camera parameters $\left(I_i(t), E_i^{k_i(t)}\right)$ can be obtained. Hence our method can generate a 3D video.

5.5 Experiment

We evaluate the algorithm from the standpoint of Reqs. 2 and 3. Firstly, we examine how well the algorithm can satisfy the requirements by a simulation and compare it with existing method which uses a set of fixed cameras. Then we demonstrate that the algorithm can be applied for the real-world environment.

5.5.1 Quantitative Evaluation by Simulation

Simulation Setup Figure 5.6 shows the arrangement of 24 active cameras and the target area for the simulation. Other resources and scenario parameters are shown in Table 5.3. We used 24 cameras to ensure the object capture by at least 8 cameras at any frame. According to the study by Starck et al. [SMN+09], 8 cameras is almost sufficient to attain 100mm reconstruction accuracy using the shape-from-silhouette algorithm, when capturing a person in a similar 3D video studio as ours. We assumed that time lags by camera motions are uniform and equal to $\tau_{\mathtt{s}}$ for all combinations of cameras and cells to be watched.

We applied our cell generation and view adjustment algorithm for the scenario and resources. The cell configuration is shown in Fig. 5.7.

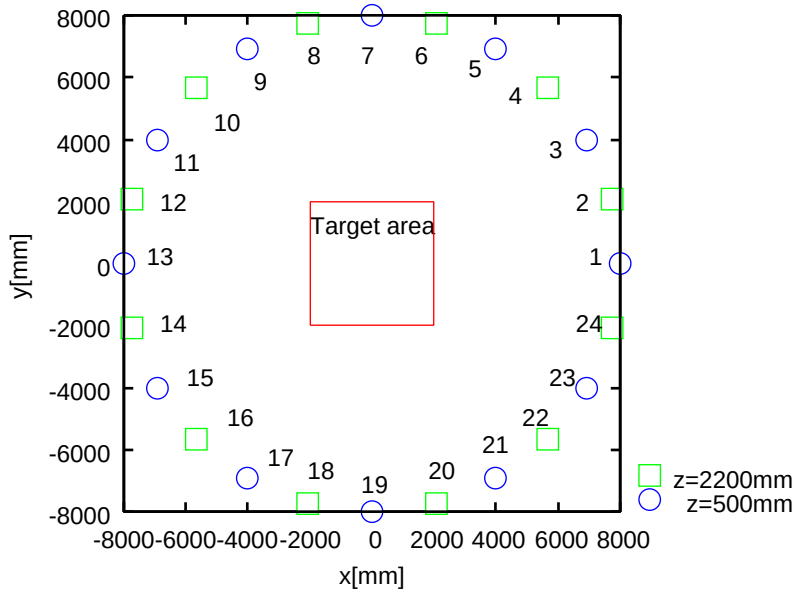


Figure 5.6: Camera configuration and target area for simulation. ©2010 IPSJ [YYNM10]

Table 5.3: Scenario and resources for simulation. ©2010 IPSJ [YYNM10]

τ_s	1.0[s]
v_{max}	300[mm/s]
s	8[mm/pixel]
B	A cylinder 900[mm] in diameter and 1800[mm] in height

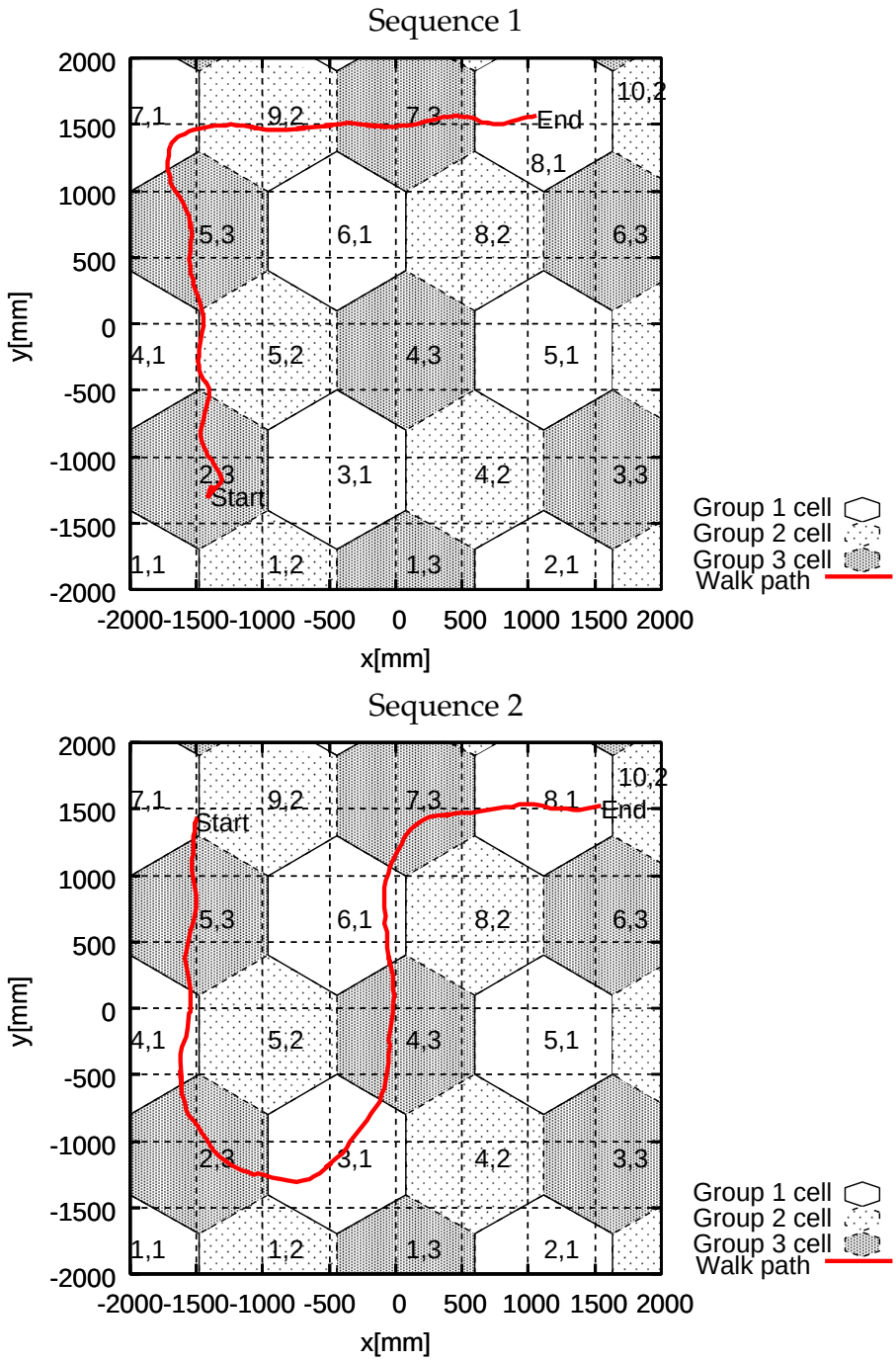


Figure 5.7: The object paths used for simulation and the cell arrangement. Each hexagon represents a cell and the numbers near the centers of them represents cell and group index. ©2010 IPSJ [YYNM10]

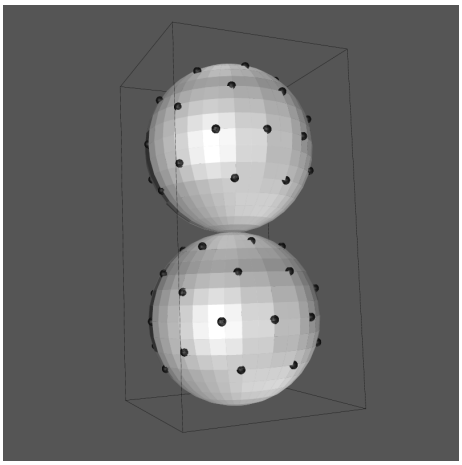


Figure 5.8: A surrogate shape model used for the analysis. Black dots represent sample points $\vec{p}_j$, which are used to compute the value of Eq. 3.7.

Baseline Performance Evaluation

Spatial distribution of Eq. (3.7) First, we evaluate how much of Req. 2 will be guaranteed by the cell arrangement. Since the subset of cameras that are guaranteed to capture the object in certain cell and relative positions of those cameras to the object depends on the position of an object, it is a function of object position in **D**. We computed the worst-case value of Eq. (3.7) using a surrogate shape model shown in Fig. 5.8. Figure 5.9 shows the value for each point in **D**.

Although the values differed between different points, it was non-zero value for all points in **D**. It indicates that all the points on the surrogate shape can always be observed by the cameras satisfying the requirements by the cell arrangement and the view adjustment result, regardless of its locomotion. The plot also reflects two characteristics of the method:

- Periodic pattern. If the distances between the object and rule borders for two camera groups are shorter than $\tau_s v_{max}$, $2N_{cam}/3$ cameras can potentially capture the object, although $N_{cam}/3$ are directed towards a cell next to the one the object is in. The value of the function tends to become larger on such locations.

- The smallest value appeared on the edge of the target area. One reason is that as the object gets closer to the cameras, the average number of cameras looking at certain point on the object lowers. This is common to existing 3D video capture methods using static cameras.

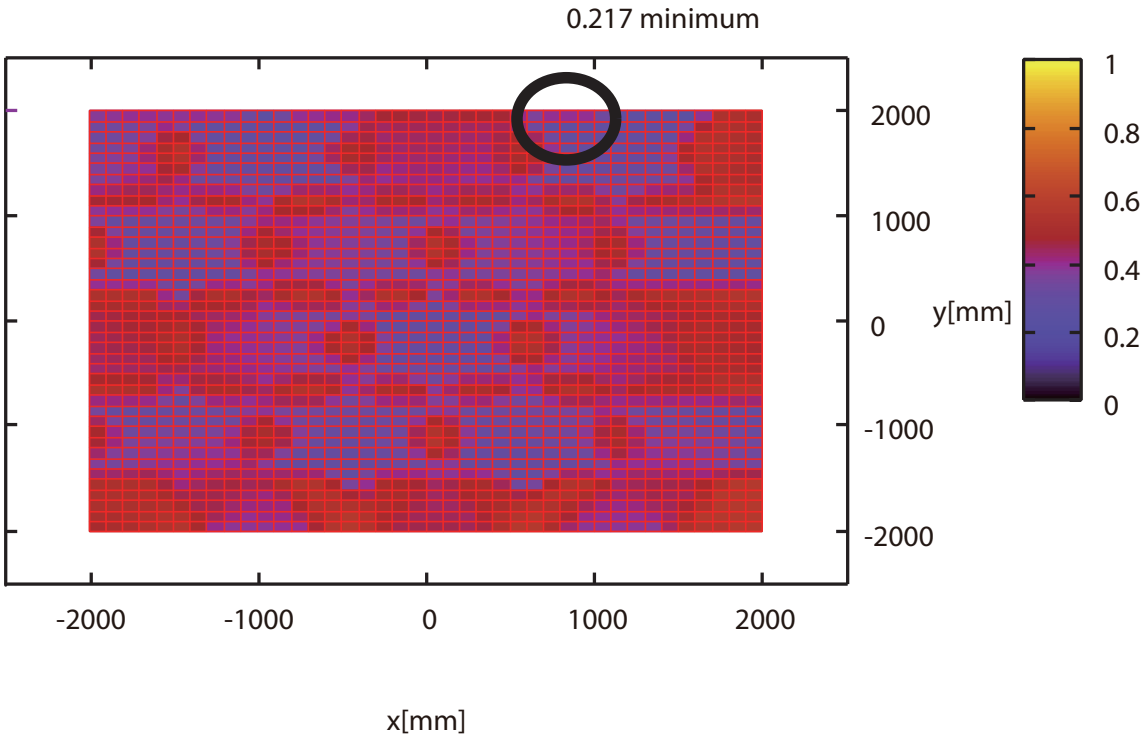


Figure 5.9: Worst-case score map for the simulation setup in section 5.5.1.

Effect of the 3-grouping algorithm Secondly, in order to examine the performance of the 3-grouping algorithm in Section 5.4.1, we compared the method with random camera grouping that divides the cameras into three groups, each of which consists of 8 cameras. We generated 1000 camera groupings randomly, and 6 groupings by our method. Then we computed the minimum values of Eq. (3.7) for every grouping.

Figure 5.10 shows the histogram of the values. The six groupings by our algorithm is designated by six arrows. We can see that the six samples based on our algorithm occupied the top 6 in this distribution. It indicates that the camera grouping algorithm was a reasonable solution.

On the contrary, the score falls to zero with most of the random groupings. It is because some points on the object could not be seen at all. Figure 5.11 shows the result by one of such examples.

Performance Comparison with an Existing Method

We compare our method with one possible capture method using static cameras, which adjusts all the camera directions and focal lengths so as to include the entire

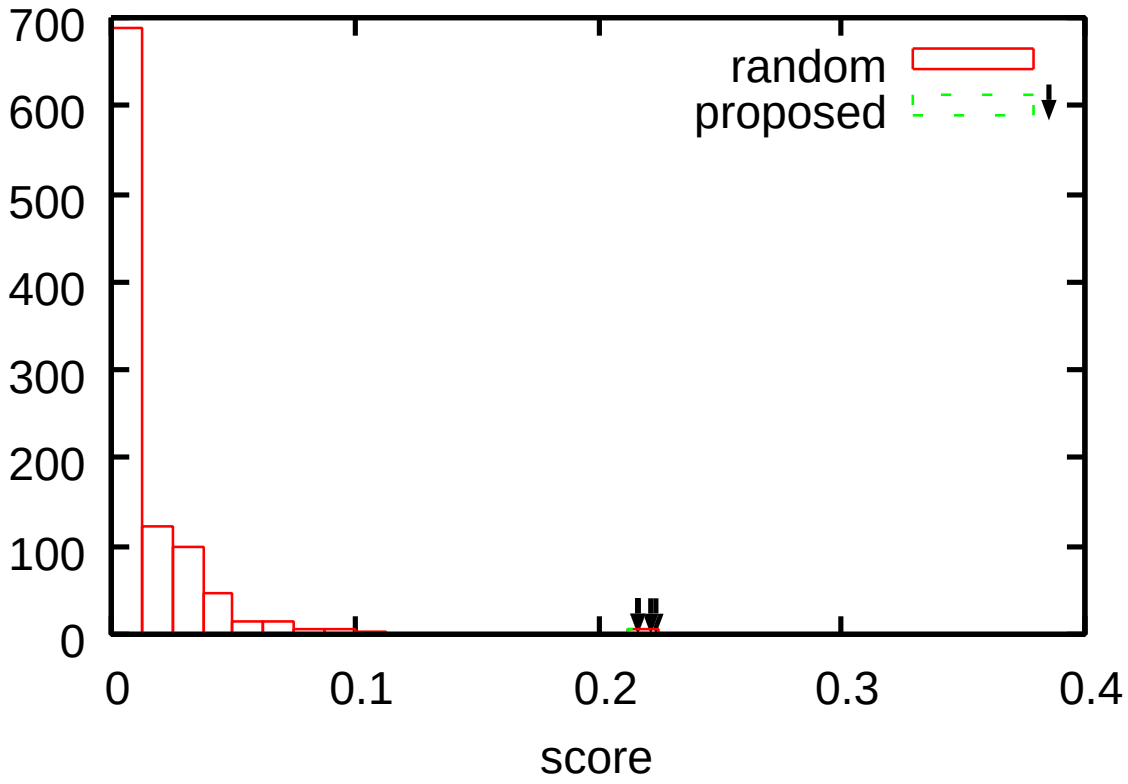


Figure 5.10: Histogram of the score by our algorithm and 1000 random group assignments. Short vertical arrows represent the six samples that used the same grouping method as our algorithm.

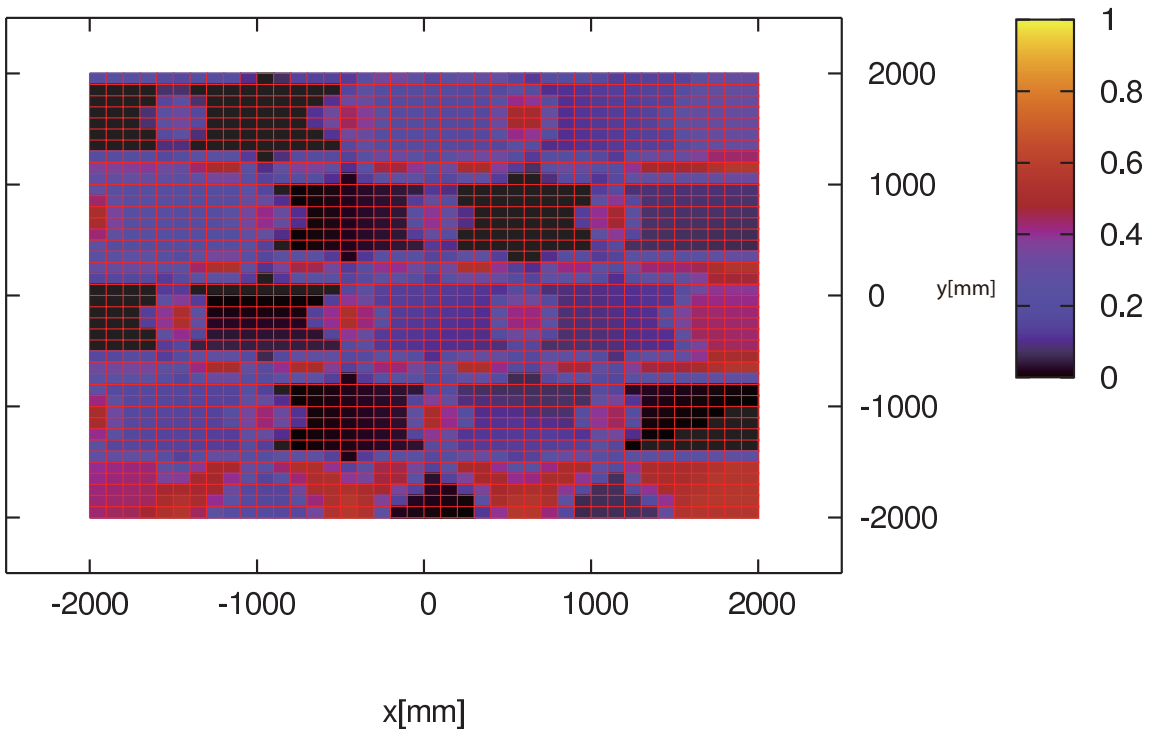


Figure 5.11: A worst-score map by one of the random groupings.

target space. We denote this method as "Fixed, wide."

We adopt these four indices for the comparison.

1. Viewpoint usage
2. Pixel usage
3. Accuracy and completeness [SCD+06] of the reconstructed shape
4. Peak signal-to-noise ratio (PSNR) of synthesized images

We used 3D video sequences of a walking person as the virtual scene to be captured. Figure 5.12 shows one of the 3D video frames. Since all of these indices are also dependent on the object motion and its trajectory, we used two different sequences. Figure 5.7 shows the object trajectories in each sequence. We also used these 3D video sequences as the ground truth shape for evaluation of the 3D shape reconstruction.

Viewpoint Usage

The active cameras were virtually controlled to capture the object and their images were synthesized by rendering the virtual scene. Then the 3D shapes of ob-

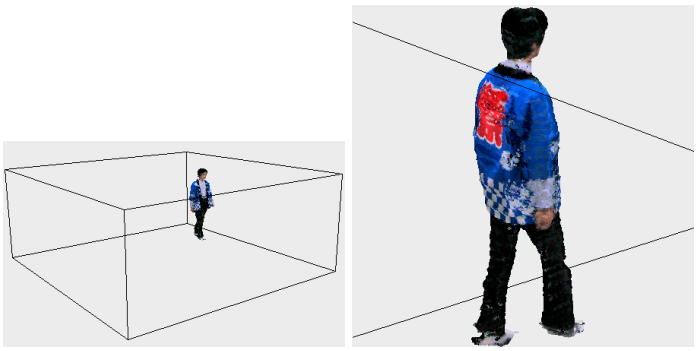


Figure 5.12: One of the frames in the 3D video sequence used for the simulation. ©2010 IPSJ [YYNM10]

jects were reconstructed from the synthesized images. We used a graph-cut based 3D shape reconstruction algorithm of [TNM08] without the super-resolution process.

We define two indices below as in Chapter 4:

$$\text{(Viewpoint usage at frame } z) = \frac{|G(z)|}{N_{\mathtt{cam}}} \tag{5.7}$$

$$\text{(Pixel usage at frame } z) = \frac{1}{|G(z)|} \sum_{i \in G(z)} \frac{N_i^{\mathtt{P}}(z)}{N_i^{\mathtt{I}}} \tag{5.8}$$

$G(z)$	$\{i \mid$camera i not in motion but captured the object at frame $z\}$
$N_i^{\mathtt{P}}(z)$	Number of pixels occupied by the object in the image of camera i
$N_i^{\mathtt{I}}$	Number of pixels in image i. e.g. 307200(=640×480) for VGA.

The camera control results for the two sequences are summarized in Fig. 5.13. It shows which cell was watched by the cameras in each group at each frame and the viewpoint usage at each frame. These figures show that the viewpoint usage exceeded 1/3 in most frames. One of the reasons is that the common views, $M_{k,l}$, overlap between two neighboring cells. Furthermore, the cameras can also observe outside $M_{k,l}$. Thus they could contribute to the reconstruction of the object shape, by partially or fully including the object in their images. The average in each sequence was 85% and 83%, respectively.

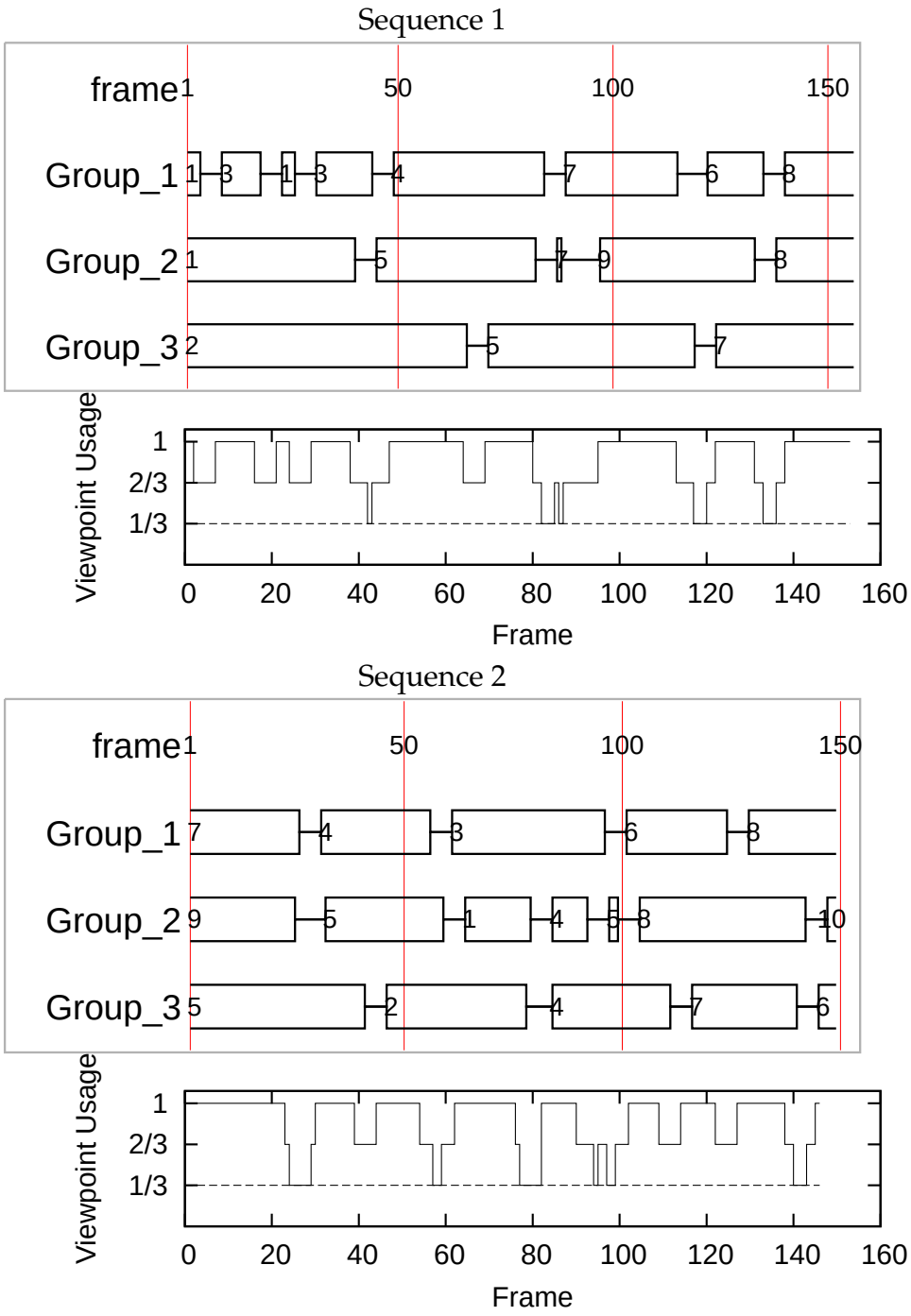


Figure 5.13: The timing chart of the active camera control for each sequence. The boxes represent the interval when the cameras in each group watched a cell. Blank part indicates that the cameras were in motion. Numbers in boxes represent the cell index k that the cameras in each group watched. ©2010 IPSJ [YYNM10]

Table 5.4: Pixel usage. Each cell shows the average ± standard deviation for the sequence. ©2010 IPSJ [YYNM10]

	Proposed[%]	Fixed[%]
Sequence 1	6.7 ± 0.32	1.9 ± 0.10
Sequence 2	7.2 ± 0.45	2.0 ± 0.16

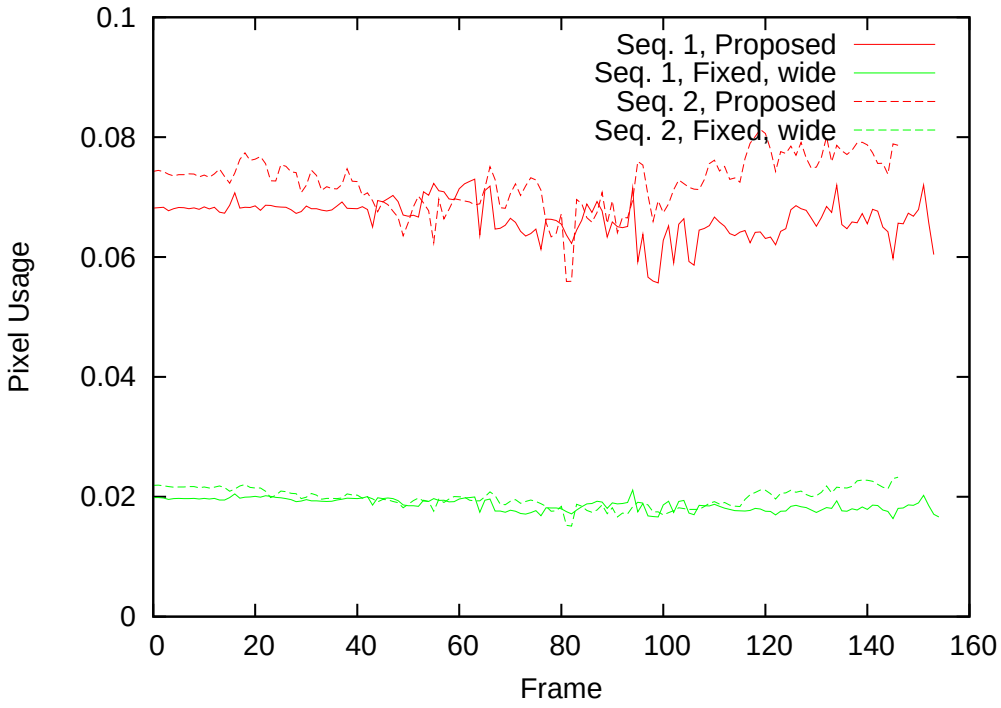


Figure 5.14: Pixel usage ©2010 IPSJ [YYNM10]

Pixel Usage

Table 5.4 and Fig. 5.14 shows the average pixel usage defined by Eq. (5.8).

The average pixel usage was 6.9% with our method and 2.0% with "Fixed, wide." The pixel usage by our method is roughly 3.5 times larger than "Fixed, wide" method. Thus the spatial resolution is about 1.8 times finer in our method in these cases.

Shape Reconstruction Accuracy and Completeness

Figures 5.15, 5.16 and 5.17 summarize the accuracy and completeness [SCD+06] of the reconstructed shapes by the two methods. 90%-accuracy means the dis-

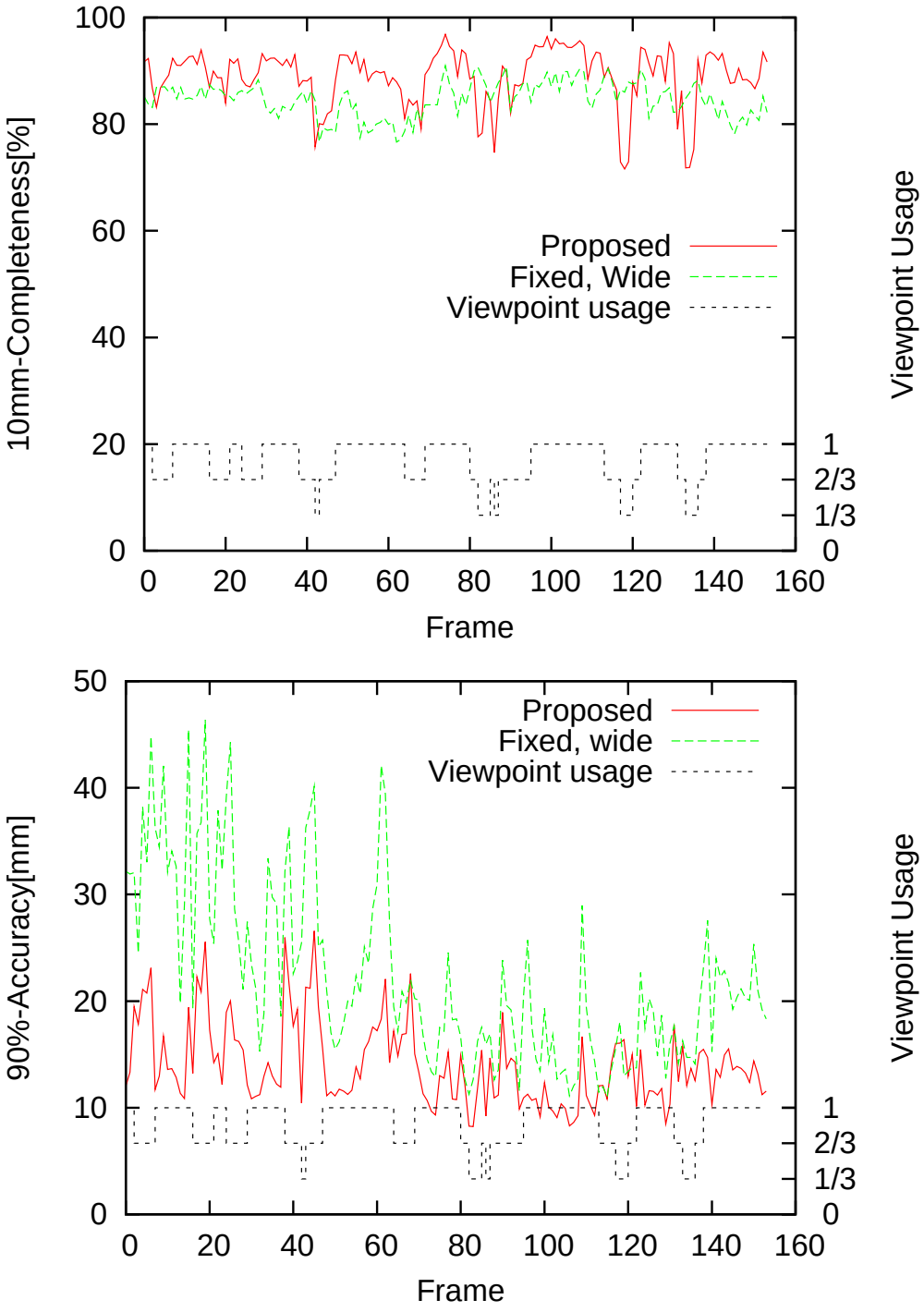


Figure 5.15: Completeness and accuracy for sequence 1. ©2010 IPSJ [YYNM10]

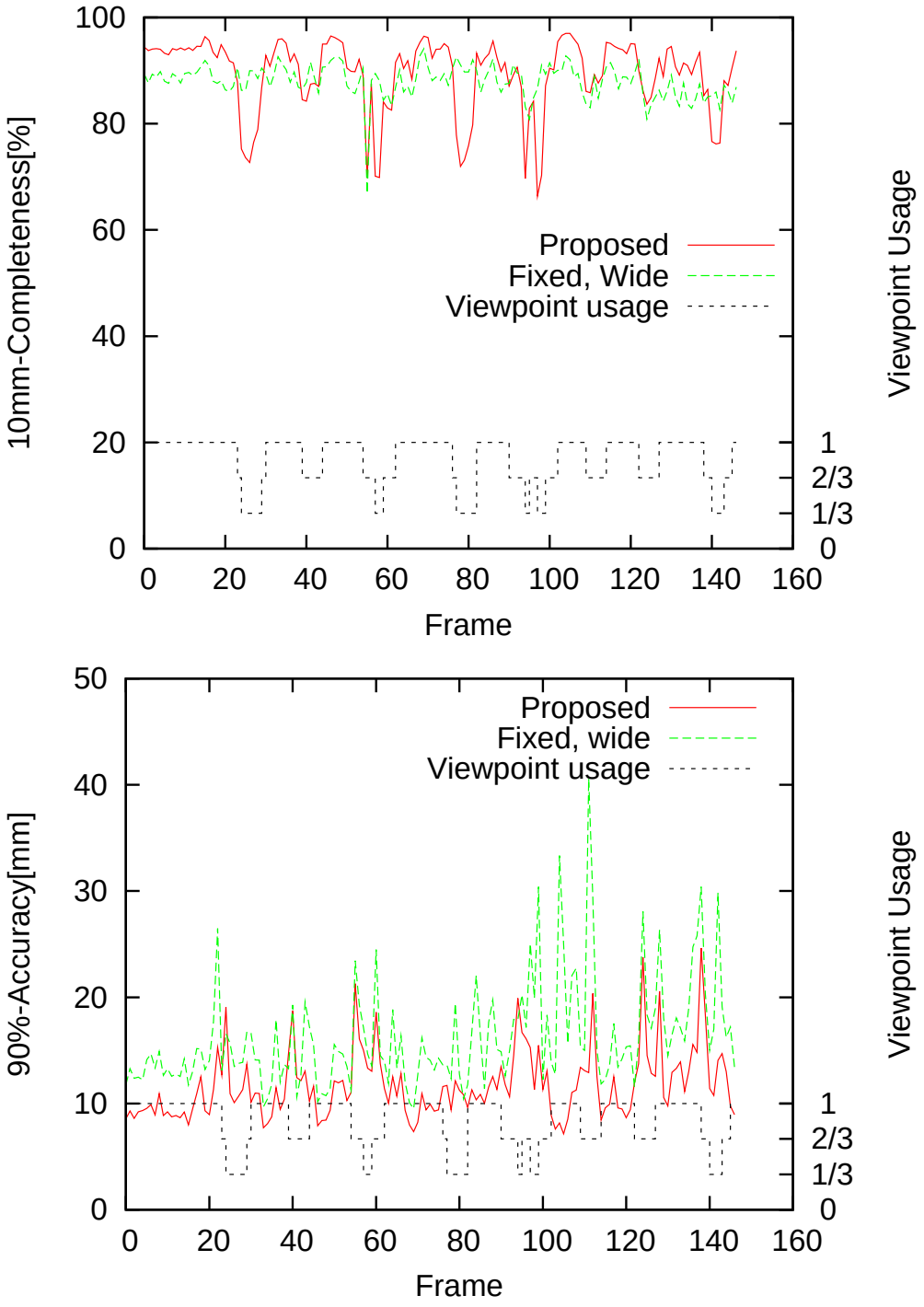


Figure 5.16: Completeness and accuracy for sequence 2. ©2010 IPSJ [YYNM10]

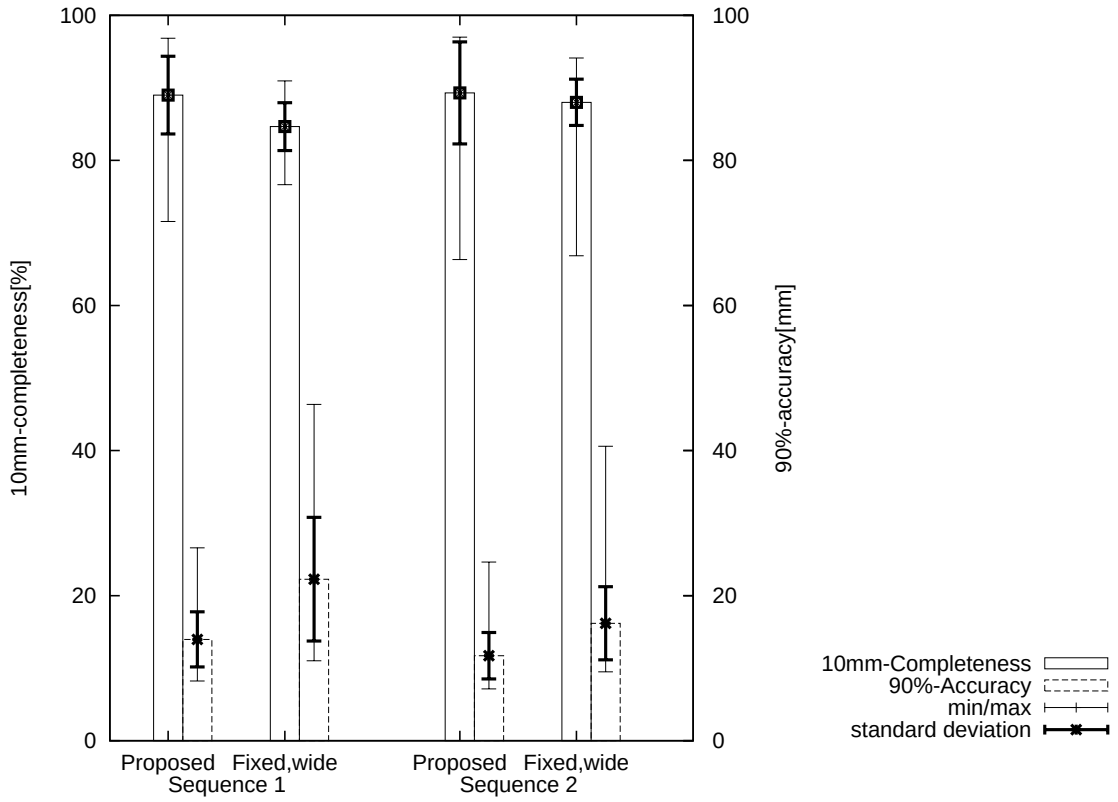
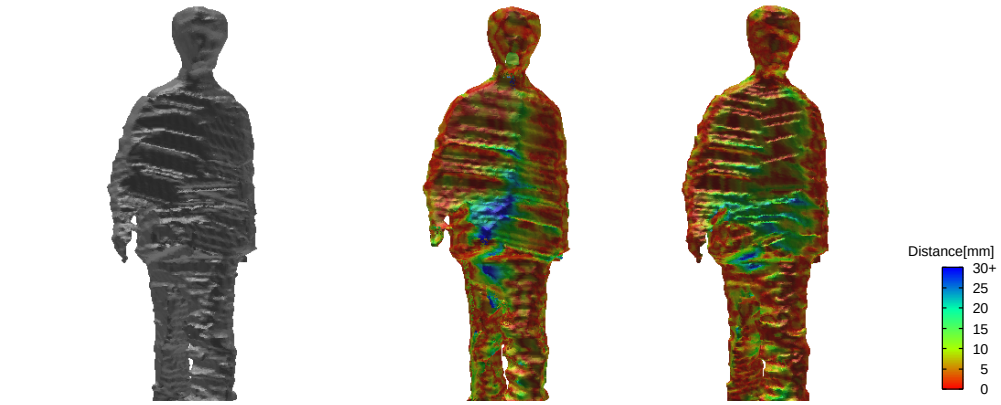


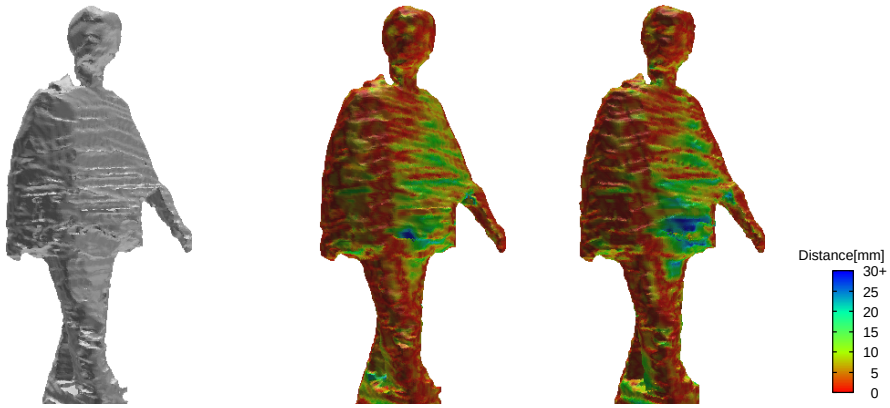
Figure 5.17: Comparison of the two methods in terms of 10mm-completeness and 90%-accuracy. ©2010 IPSJ [YYNM10]



	Ground truth	Proposed 8 cameras	Fixed, wide 24 cameras
90%-accuracy[mm]		12.1	14.7
10mm-completeness[%]		71.9	85.9

PSNR[dB]		22.8	21.3

Figure 5.18: Shape reconstruction result in frame 135 of sequence 1. The first row shows the shapes, whose colors indicate the distance to the ground truth surface. Distances are computed using Metro[CRS98]. The bottom row shows the rendered images and their PSNR in comparison to the ground truth. ©2010 IPSJ [YYNM10]



Ground truth	Proposed 16 cameras	Fixed, wide 24 cameras
90%-accuracy[mm]	9.9	11.5
10mm-completeness[%]	92.1	87.9

PSNR[dB]	23.8	20.4

Figure 5.19: Shape reconstruction result in frame 95 of sequence 1. ©2010 IPSJ [YYNM10]

Proposed Fixed, wide

Figure 5.20: Images obtained by camera 1 at frame 95 in sequence 1. ©2010 IPSJ [YYNM10]

tance d such that 90% of the reconstructed surface is within d millimeters of the ground truth, and 10mm-completeness is the percentage of the reconstructed surface that are within 10mm of the ground truth. On average, our method performed equally well or better than the "Fixed, wide" method.

At the frames where the viewpoint usage dropped to 1/3, the completeness of our method lowered significantly but the accuracy was not degraded. In other frames, where more than 2/3 of the viewpoints were available, our method outperformed the "Fixed, wide" method in both measures. Fig. 5.18 shows the reconstructed shape at frame 135 in sequence 1. Our method resulted in poorer completeness due to the lower number of viewpoints. We can see a larger error at the chest part of the object with our method. This part was poorly reconstructed due to self-occlusion. However, the accuracy as a whole was better than the "Fixed, wide" method. Fig. 5.19 shows the frame where our method resulted in better accuracy and completeness despite the smaller number of viewpoints. One possible reason to describe both results is that, as long as the object surface can be observed by multiple cameras, stereo matching can be performed more accurately with the help of finer texture cues obtained by our method. As shown in Fig. 5.20, "Fixed, wide" method produced smaller images of the object, affecting the shape reconstruction accuracy.

PSNR of Synthesized Images

Finally, we evaluate the appearance of generated 3D videos. For each frame, a virtual viewpoint was set up in front of the object. The viewpoint was located three meters away from the object and its angle of view was set to 32 degrees, with which the entire object can be observed in its view. The original and the

reconstructed models were rendered and the images were compared to compute the PSNR for each frame. The results are summarized in Fig. 5.21. Our method resulted in the higher PSNR at every frame. The images for frames 135 and 95 in sequence 1 are shown in Fig. 5.18 and Fig. 5.19. At frame 135, although the shape reconstruction completeness is not as good as the "Fixed, wide" method, our method resulted in better image appearance. One of the reasons is that the "Fixed, wide" method generated coarser textures.

Conclusion

In our method, the number of viewpoints decreases to 1/3 for the worst frame. Lack of viewpoints sometimes leads to poorer completeness compared to the "Fixed, wide" method. However, our method can maintain the accuracy of 3D video even with those frames, by capturing object surfaces in higher spatial resolution. Additionally, it also provides finer textures on the object. Having these advantages, the overall accuracy of the 3D video and its appearance were significantly improved.

5.5.2 Studio Experiment

We have also tested our method in a physical setup. We arranged 23 active cameras in our 3D video studio, which is about 8 meters square. The studio and the camera setup are shown in Fig. 5.22. Each active camera consists of a zoom camera SONY DFW-VL500 and a PTU-46 pan-tilt unit by Directed Perception, Inc. We set up 23 computers as camera nodes and 1 computer as the master node. All the nodes were connected by Ethernet and communication between the nodes were implemented by UDP. The system clocks were synchronized by NTP.

We captured a stuffed toy on a radio control car as shown in Fig. 5.23. We have set scenario parameters as shown in Table 5.5 and Fig. 5.22. Note that the object and the parameters were chosen reflecting the limitations of our studio equipment, not our algorithm itself. We chose a small object because the studio was not large enough and the cameras were too close to capture a whole body of a person. A capture system for a walking person using our algorithm can be realized by scaling up the studio, target area, $v_{\mathtt{max}}$, $\mathbf{B}$ and s. For example, if we set up a studio 4 times larger than ours, the cameras can be placed 4 times farther and it would enable the capture of a person that walks at 832[mm/s]. The target area size is 12m $\times$ 8m, and a spatial resolution of 20[mm/pixel] can be achieved.

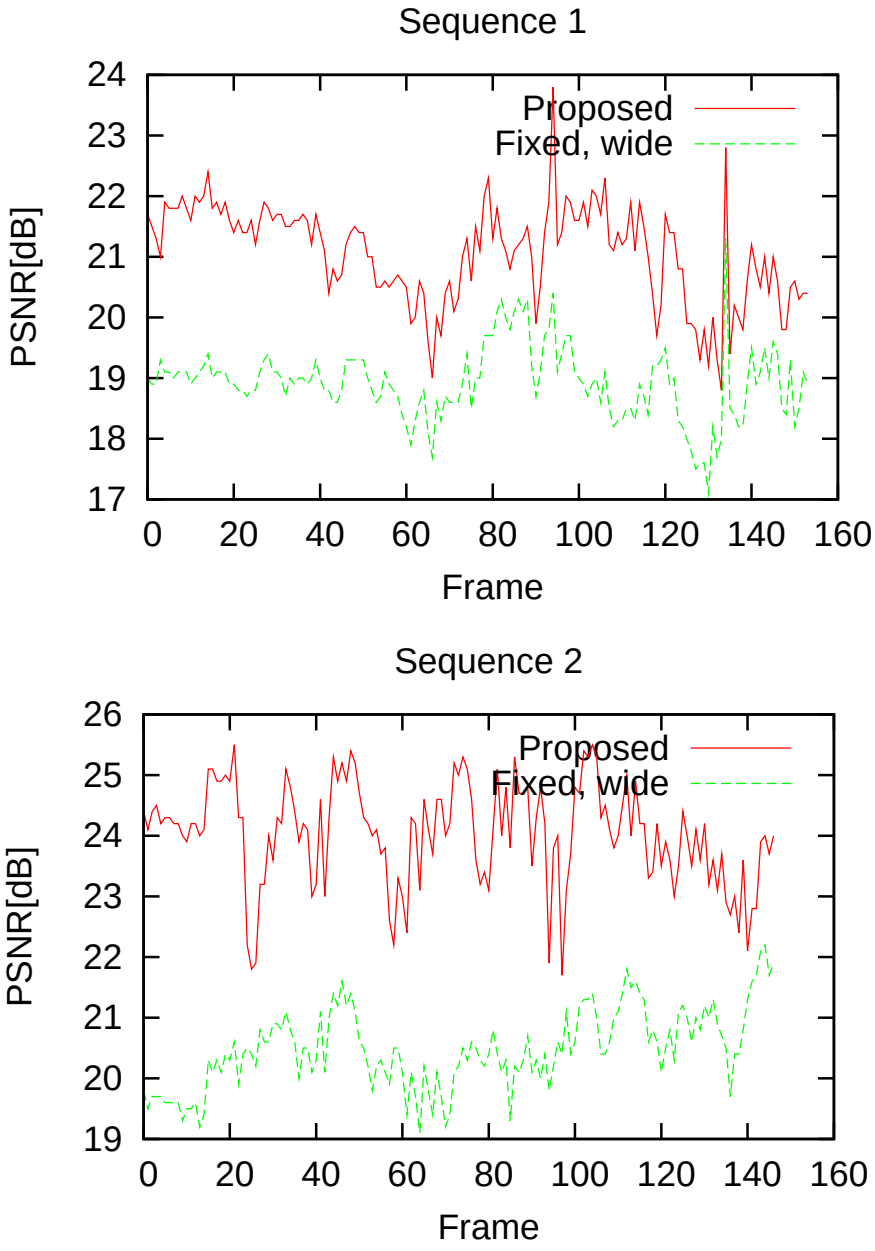


Figure 5.21: PSNR of rendered images. ©2010 IPSJ [YYNM10]

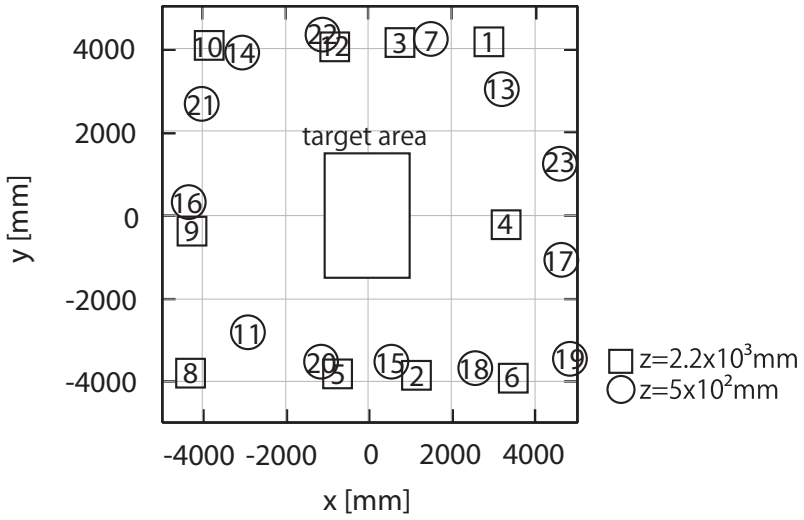


Figure 5.22: Our studio, camera setup and target area. The numbers in circles or squares represent the camera positions. The cameras represented by the squares were mounted on the stations hung from the ceiling. The others were mounted on the stations on the floor. ©2010 IPSJ [YYNM10]

Figure 5.23: The object captured in the real studio experiment: a stuffed toy on a radio control car. ©2010 IPSJ [YYNM10]

Table 5.5: Scenario and resources for studio experiment. ©2010 IPSJ [YYNM10]

τ_s	1.2[s]
v_{max}	208[mm/s]
s	5[mm/pixel]
B	A cylinder 800[mm] in diameter and 500[mm] in height

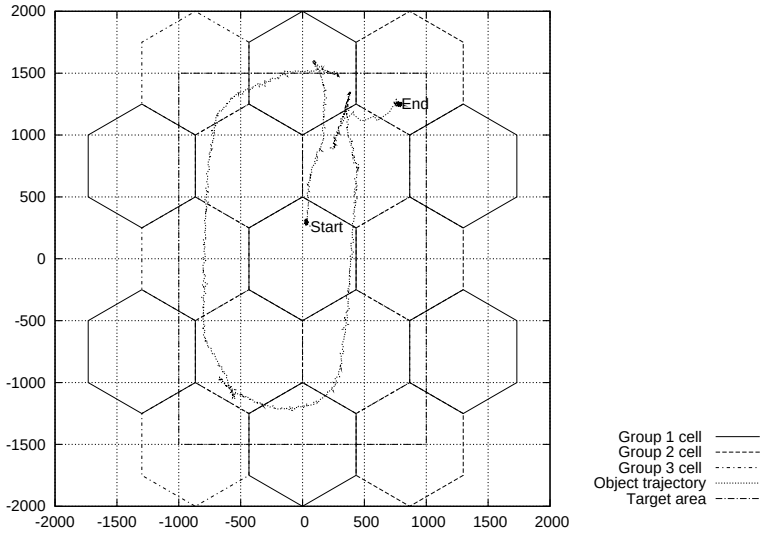


Figure 5.24: Generated cells and object trajectory in the studio experiment. ©2010 IPSJ [YYNM10]

More generally, the scalability of our method can be discussed as follows: Assume that the target area size is scaled by α and target object speed by β. According to Eq. (5.5), cell size must be scaled by β. Consequently, spatial resolution will be reduced by β^{-1}. Number of cells will be scaled by $\alpha^2\beta^{-2}$; since the cells must fill the entire target area, number of the cells increases in proportion to α^2, but reduced by β^{-2} because the area of each cell is scaled by β^2.

Generated cells are shown in Fig. 5.24. 17 cells were generated in total. The number of camera control parameters is shown in Table 5.6. For example, the first group consists of eight active cameras being assigned to seven cells, which produces $8 \times 7 = 56$ virtual fixed cameras. In this case, we had 131 virtual fixed cameras in total. We calibrated the cameras in two steps. Firstly, intrinsic parameters were estimated by Zhang's method[Zha00] for the 131 virtual fixed cameras independently. Secondly, the extrinsic parameters were estimated by the eight-

Table 5.6: Number of the cameras, cells and virtual fixed cameras for each group. ©2010 IPSJ [YYNM10]

l	$\|\|\mathbf{A}_l\|\|$	$N_l^{\mathtt{cell}}$	$\|\|\mathbf{A}_l\|\| \times N_l^{\mathtt{cell}}$
1	8	7	56
2	8	5	40
3	7	5	35
Total	23	17	131

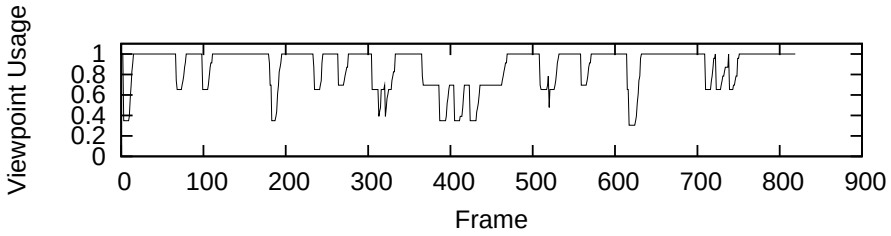
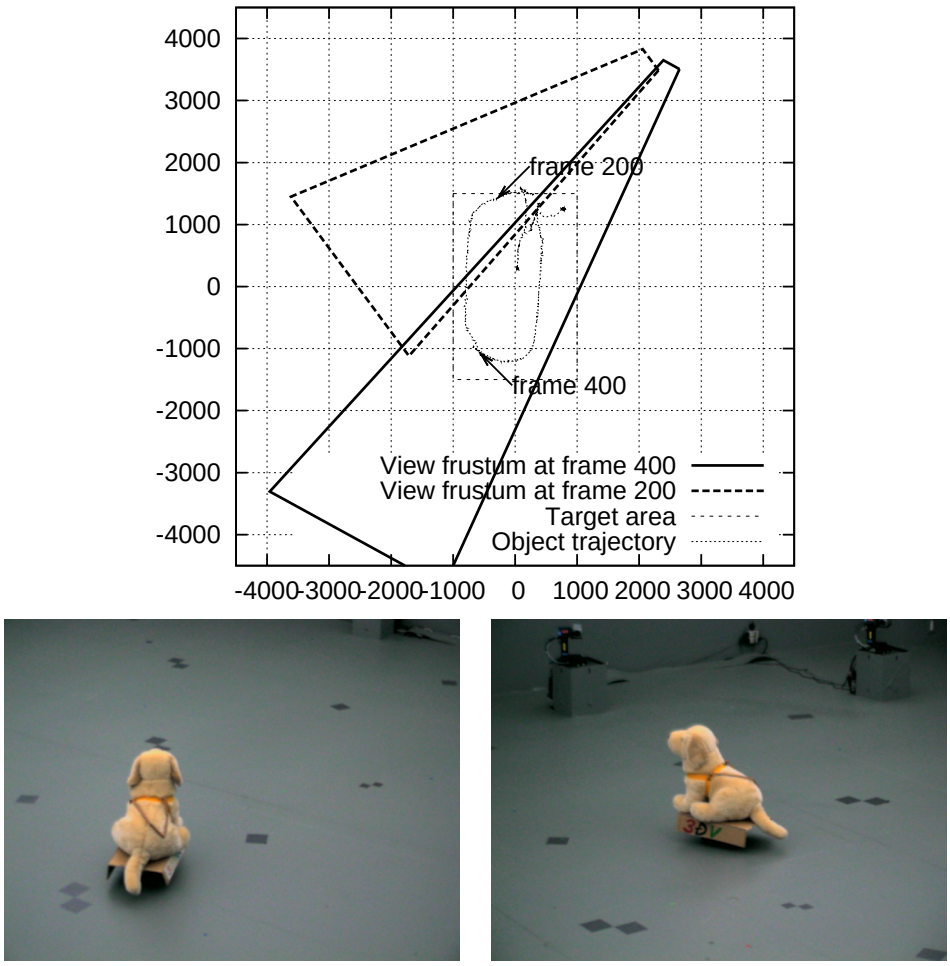


Figure 5.25: Viewpoint usage in the studio experiment. ©2010 IPSJ [YYNM10]

point algorithm [HZ04] for each pair of cameras with 2D-to-2D correspondences of unknown 3D points [1], and then refined through a bundle adjustment process which minimizes the sum of symmetric epipolar distances of all the cameras. At this step, all the virtual fixed cameras corresponding to the same cell can be calibrated simultaneously. Moreover, the virtual fixed cameras corresponding to three adjacent cells surrounding a vertex can be calibrated simultaneously to reduce the calibration work load. Thus the number of extrinsic calibration tasks required in total depends on the cell arrangement. At least $(\max_l N_l^{\mathtt{cell}})$ calibration tasks are required. Some additional calibration tasks are needed for unifying the world coordinate systems by the result of each calibration process. In our case we performed this step by 10 calibration tasks for the different combinations of the cells.

The object moved on the path shown in Fig. 5.24 at varying speeds that is slower than $v_{\mathtt{max}}$, and was captured by our algorithm. Figure 5.25 shows the viewpoint usage. The object was captured by 7 or more viewpoints at any frame. Since the object moved back and forth, it traveled across rule borders several times around frame 400. The viewpoint usage is low because the cameras in the two groups switched their views several times between cells in order to follow the object. During this period, the other one group could capture the object con-

[1] We adopted a similar implementation with Svoboda's[SMP05]. We captured a moving point light source as a multi-view video in order to obtain 2D-to-2D correspondences robustly.



frame 200 frame 400

Figure 5.26: Captured images by camera 1, object positions and camera view frustums at frames 200 and 400. ©2010 IPSJ [YYNM10]

tinuously. Another difference from the simulation in section 5.5.1 is that some cameras could finish switching its view in less than τ_s seconds. Even in this case, our algorithm can guarantee that the object is continuously captured with more than $\lfloor N_{\mathtt{cam}}/3 \rfloor$ cameras. Figure 5.26 shows some examples of captured images with object positions. We can see that the object size in images is almost uniform regardless of the distance from the camera. Finally, the object shape in each frame was reconstructed using a graph-cut based 3D shape reconstruction algorithm of [TNM08] without the super-resolution process. Figure 5.27 shows a rendered example of the reconstructed 3D video. We can see that fine details, such as the harness or letters written on the car, are successfully reconstructed.

Figure 5.27: Reconstruction result at frame 395 with 8 cameras. ©2010 IPSJ [YYNM10]

5.6 Summary

We have proposed a cell-based 3D video capture method that can capture a freely-moving object. Our method adjusts camera views in off-line process before capturing, then controls cameras based on it. It enables accurate camera calibration and can ensure that the object is captured using at least one third of the cameras when possible with given resources and scenario. We have shown our method can capture a moving object with higher resolution compared to an existing method with static cameras.

Chapter 6

Cell-Based 3D Video Capture System for a Figure Skater

The experiments in Chapters 4 and 5 have demonstrated the effectiveness of the cell-based algorithms in laboratory environment. In order to show that our cell-based method can be applied for a more practical situation, we design a figure skater recording system for an imaginary ice skate arena based on the algorithm described in Chapter 5.

Chapters 4 and 5 focused on the control of active cameras based on the motion of the object, but not positions of active cameras. These algorithms required that active camera positions are given a priori. Additionally, the algorithm described in Chapter 5 required that the upper limit of time needed to change active camera state was given as a constant value. However, in practice, we would need to know requirements about motion characteristics and layout of the active cameras in order to design and build a 3D video capture system before applying the algorithm. In order to fill this gap, we will build a more precise active camera model based on typical existing PTZ camera and analyze how it affects the cell-based 3D video capture algorithm.

6.1 Problem Description

We design a 3D video capture system that can record a figure skater based on the cell-based 3D video capture algorithm described in Chapter 5. Figure 6.1 and 6.2 shows an imaginary ice skate arena we consider in this chapter. A skater moves on the ice rink shown in Fig. 6.2.

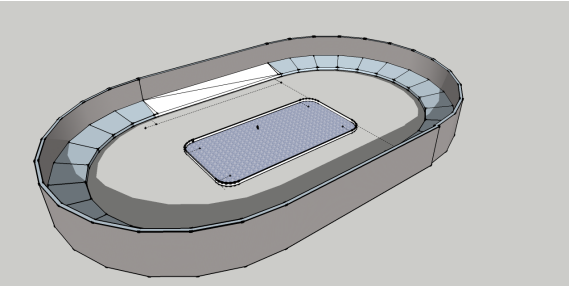


Figure 6.1: A sketch of an imaginary ice skate arena we consider in this chapter.

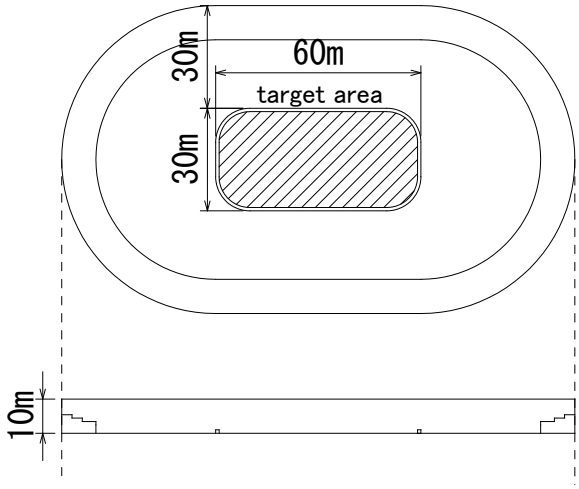


Figure 6.2: The plan of the ice skate arena. According to the regulations by the International Skate Union [Int08], an ice rink for Short Program and Free Skating is rectangular and approximately 60×30 meters.

Due to the physical constraint of the arena, we will mount all active cameras on the ceiling of the arena, which is 10 meters above the floor The remaining problem is how to decide the locations within the plane to mount the active cameras.

6.1.1 Scenario

The target area, **D**, is designated in Fig. 6.2. The skater moves within the area at $v_{\mathtt{max}} = 10$[m/s] at maximum. We set the spatial resolution requirement to $s = 8$[mm/pixel].

We adopt a cylinder shown in Fig. 6.3 as a bounding volume. The radius of the cylinder is one meter and its height is two meters.

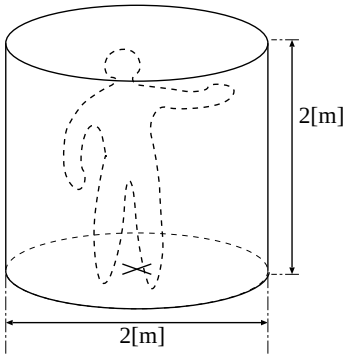


Figure 6.3: Bounding volume used for the figure skater capture system.

6.1.2 Resources

We adopt SONY EVI-HD7V pan/tilt/zoom camera as a typical example of off-the-shelf active camera. The specifications of the camera are shown in Table 6.1. EVI-HD7V camera doesn't have synchronization signal input. Therefore we cannot actually build a 3D video capture system with them. However, we can use the values of mechanical and optical characteristics to get a realistic reference of off-the-shelf active camera.

Table 6.1: Specifications of SONY EVI-HD7V pan/tilt/zoom camera

image sensor	type 1/3 (4.8mm×3.6mm), CMOS
image resolution	1920 × 1080 [1]
pan angle	±100 degrees
frame rate	60 frames/sec.
tilt angle	±25 degrees
pan speed	300 degrees/sec
tilt speed	125 degrees/sec
horizontal angle of view	70(wide) – 8(tele) degrees
focal length	3.4(wide) – 33.9(tele) millimeters
zoom speed	30.5 millimeters/sec

[1] The camera supports several output signal formats and image resolution. We choose the highest resolution here.

EVI-HD7V is controlled by command packets. We use four necessary motions for the cell-based tracking; pan, tilt, zoom and focus. It is realized by two packets, which are "Pan/tilt" and "Zoom/focus" commands. The two command pack-

ets must be transmitted to the camera separately. The camera can receive only one command per each video frame. Moreover, it requires one video frame to send back an acknowledge packet after receiving a command packet. In total, the transmission of a command requires three video frames whenever changing the views of a camera from a cell to another cell.

The pan, tilt and zoom motion speeds of the camera are shown in Table 6.1.

6.1.3 Constant-speed camera motion duration modeling

We assume that the angular speed for each axis can be regarded as constant during its motion, so that the time needed to finish a pan or a tilt motion increases in a direct proportion to the angular differences. Similarly, we assume that the time needed to finish a zoom motion increases in a direct proportion to the difference between current and target focal lengths. We define some symbols in Table 6.2 for the following discussion.

Based on the assumptions, we model the delay by each motion as

$$\tau_{\mathtt{pan}}(\Delta\theta_{\mathtt{pan}}) = \tau_{\mathtt{base}} + \frac{\Delta\theta_{\mathtt{pan}}}{\omega_{\mathtt{pan}}} \tag{6.1}$$

$$\tau_{\mathtt{tilt}}(\Delta\theta_{\mathtt{tilt}}) = \tau_{\mathtt{base}} + \frac{\Delta\theta_{\mathtt{tilt}}}{\omega_{\mathtt{tilt}}} \tag{6.2}$$

$$\tau_{\mathtt{zoom}}(\Delta f) = \tau_{\mathtt{base}} + \frac{\Delta f}{\dot{f}_{\mathtt{zoom}}}, \tag{6.3}$$

and the overall delay by

$$\tau_{\mathtt{s}}(\Delta\theta_{\mathtt{pan}}, \Delta\theta_{\mathtt{tilt}}, \Delta f) = \max(\tau_{\mathtt{pan}}, \tau_{\mathtt{tilt}}, \tau_{\mathtt{zoom}}). \tag{6.4}$$

Where $\tau_{\mathtt{base}}$ is a time duration between object movement that causes camera movement and beginning of the camera motions. We assume $\tau_{\mathtt{base}} = 0.1[\mathrm{s}]$, which includes processing time and command transmission which takes 3 video frame intervals ($1/60 \times 3 = 0.05[\mathrm{s}]$).

Table 6.2: Symbols and their values for the scenario given in this chapter.

Symbol	Value	Unit	Description
$v_{\mathtt{max}}$	10	[m/s]	Maximum speed of the object.
s	8	[mm/px]	Lowest allowable spatial resolution.
W	1920	[px]	Number of pixels in a image row.
w	4.8	[mm]	Effective width of an image sensor in the cameras.
r	1	[m]	Radius of bounding volume cylinder.
$\dot{f}_{\mathtt{zoom}}$	30.5	[mm/s]	Maximum zoom speed of the active cameras.
$\omega_{\mathtt{pan}}$	300	[deg/s]	Maximum pan speed of the active cameras.

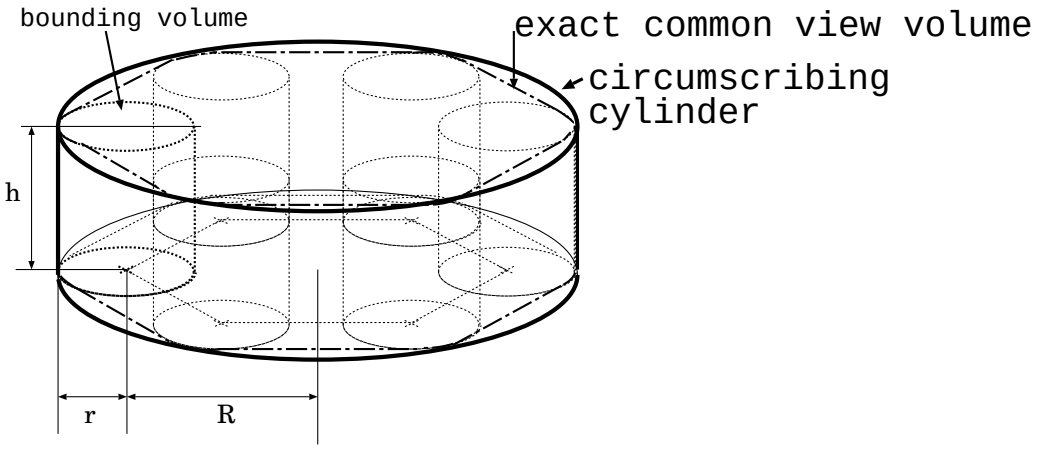


Figure 6.4: The common view volume is approximated by a circumscribing cylinder in this chapter. The radius of the cylinder is $R + r$ and the height is the same to the bounding volume.

6.2 Camera motion constraints by Cell-Based 3D Video Capture Algorithm

By substituting $\tau_{\mathtt{s}}$ in Eq. (5.5) with Eq. (6.4), we will get relationship between cell size, camera characteristics and camera locations. We only need to consider the cases where a camera undergoes the largest motion for each axis, and when the object moves at the maximum speed.

For simplicity, we approximate the common view volume by a circumscribing cylinder as shown in Fig. 6.4. Due to this approximation, derived conditions are not necessary conditions but sufficient conditions.

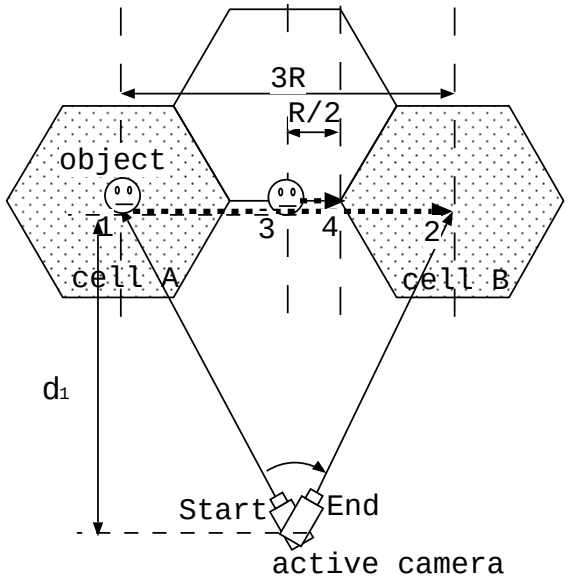


Figure 6.5: Two cell arrangement that maximizes camera's pan motion.

6.2.1 Lower Limit of the Cell Size by Camera Motion

First we consider the pan motion. Let d be the horizontal distance from a camera to the center of a cell and the maximum pan speed be $\omega_{\mathtt{pan}}$. Note that, since pan axis of each camera is perpendicular to the floor plane, the pan motion is determined by the projection points of the camera and the object to the floor plane.

The maximum pan motion occurs when the object moves in the orthogonal direction to a camera's viewing direction and two cells are in the equal distance from the camera, as shown in Fig. 6.5. In this case, the pan angle is changed by

$$2\tan^{-1}\frac{3R}{2d_1}. \tag{6.5}$$

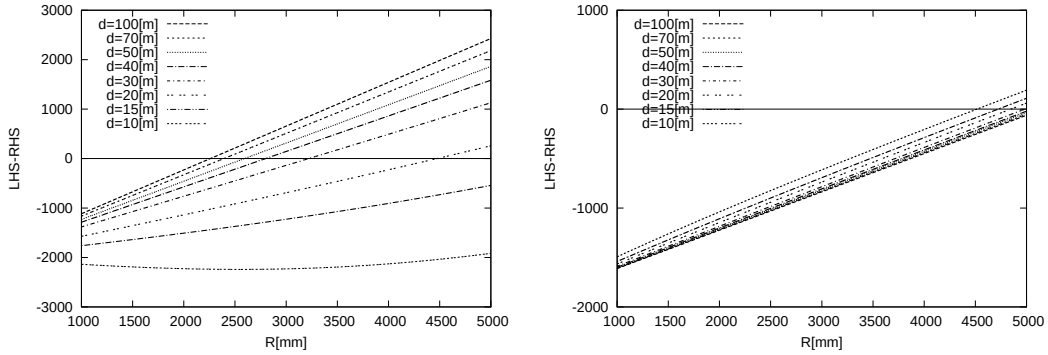
Since $d_1 < d$,

$$R \geq 2\left(\frac{2\tan^{-1}\frac{3R}{2d}}{\omega_{\mathtt{pan}}} + \tau_{\mathtt{base}}\right) v_{\mathtt{max}} \tag{6.6}$$

gives a sufficient condition of R and d by the pan motion.

Next we consider zoom motion. Focal length for the camera to capture the farthest point in the cell with spatial resolution s [mm/pixel] is given by

$$f = \frac{\sqrt{(d+(R+r))^2+h^2}w}{sW} \tag{6.7}$$



(a) (6.6):for pan motion

(b) (6.9):from zoom motion

Figure 6.6: Camera control constraints by (6.6) and (6.9). The vertical axis denotes the difference between LHS and RHS of each inequality. Thus the inequalities are satisfied in the positive value area.

where W is number of the pixels in an image row, w is the effective width of the image sensor in millimeters, and h is the height of the camera. Note that we use 3D distance from the camera to the object here. The largest motion occurs when the projection of a rule border to the floor plane is orthogonal to that of the optical axis of a camera. When the object moves between such cells, the focal length of the camera should be changed by

$$\Delta f = \frac{(\sqrt{(d+(R+r)+3R)^2+h^2} - \sqrt{(d+(R+r))^2+h^2})w}{sW} \tag{6.8}$$

to keep the spatial resolution. By substituting $\tau_{\mathtt{s}}$ in Eq. (5.5) with Eqs. (6.3) and (6.8), we get

$$R \geq 2v_{\mathtt{max}} \left(\frac{w(\sqrt{(d+4R+r)^2+h^2} - \sqrt{(d+R+r)^2+h^2})}{sW\dot{f}_{\mathtt{zoom}}} + \tau_{\mathtt{base}} \right). \tag{6.9}$$

Figure 6.6 shows the difference between LHS and RHS of (6.6) and (6.9). Each inequality is satisfied where the value is positive. Under the condition that $1000 \leq R \leq 50000$ and $d \geq 15000$, the both difference values are non-decreasing functions of d, which means that the inequalities only limit the minimum value of R.

Same discussion can be held for the tilt motion, however, we can omit it because it is not a limiting factor in the case discussed in this chapter. As will be described in section 6.3, we will place active cameras more than 15 meters apart from the target area by the horizontal distance. Tilt motions will be much smaller

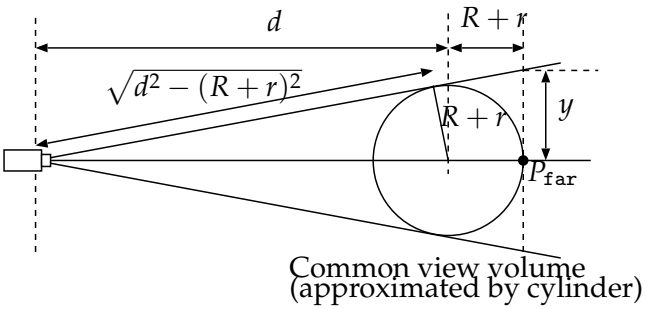


Figure 6.7: Common view volume of a cell and its view frustum.

than pan motions with such configuration.

6.2.2 Upper Limit of the Cell Size by Spatial Resolution

The algorithm described in Chapter 5 adjusts camera views to include a common view volume of each cell. Spatial resolution should be kept finer than s at the farthest points in common view volumes of every cell. Figure 6.7 shows a view frustum of a camera which uses the longest focal length that can cover the entire common view volume. At point $P_{\mathtt{far}}$, the physical length $2y$ is sampled by W pixels or less. The value of y can be expressed as:

$$y = \frac{d + (R + r)}{\sqrt{d^2 - (R + r)^2}}(R + r), \tag{6.10}$$

Thus the spatial resolution here cannot be higher than

$$\frac{2y}{W} = \frac{2(d + (R + r))(R + r)}{W\sqrt{d^2 - (R + r)^2}}. \tag{6.11}$$

Since the resolution must be finer than s,

$$s \geq \frac{2(d + (R + r))(R + r)}{W\sqrt{d^2 - (R + r)^2}}. \tag{6.12}$$

When the value of d is fixed, the RHS of Eq. (6.12) is non-decreasing function of R as shown in Fig. 6.8 . Therefore the inequality gives the upper bound of R.

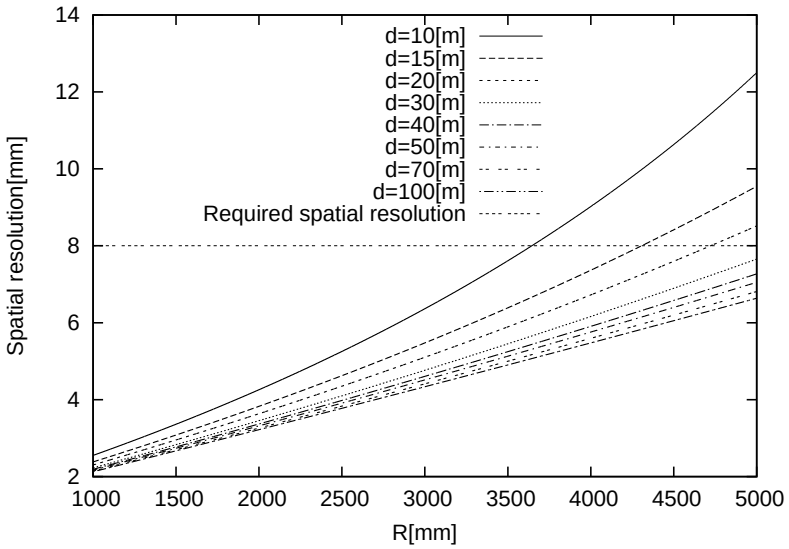


Figure 6.8: Spatial resolution derived from cell size. The condition is satisfied where the value is below the "Required spatial resolution" line.

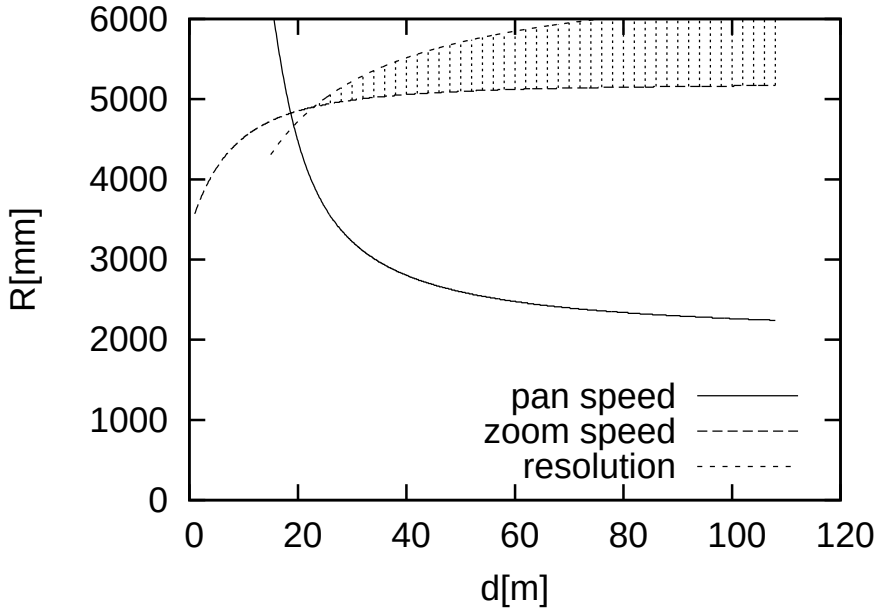


Figure 6.9: Constraints on R and d. Hatched area designates the feasible region.

6.2.3 Upper Limit of Distance from Cameras to Cells by Zoom Limit

Since the cameras have the upper limit of focal length, there is also an upper limit of the distance from a camera where we can satisfy the spatial resolution requirement. According to Eq. (6.7), the horizontal distance from object to every camera must be shorter than 108.0 meters.

On the other hand, the lower limit of the focal length will not be a limiting factor in our application. For example, if the focal length is set to the shortest, which is 3.4 millimeters, the spatial resolution will be satisfied only within 10.9 meters from the camera. As discussed in the next section, we will place cameras farther than that and use longer focal lengths.

6.3 Camera Layout and Cell Arrangement

We basically follow a heuristic method adopted in existing 3D video studio design methodology for viewing directions of the cameras. The arena and the target area is symmetric. In order to maintain the 3D video reconstruction equally well regardless of the object's shape, direction and the position, the cameras should be distributed in all the directions from the center of the target area. However, in our case, the distribution of viewing directions to the object changes when the object gets closer to the cameras. It changes the baseline length between cameras, and sometimes gives bad impact on Req. 2. Thus the cameras should be placed as distant as possible from the target area. As an intuitive solution, we will place one active camera each on the radial lines designated in Fig. 6.10 and try to maximize the distance from the target area within the constraints.

We got necessary conditions about the cell size and distance from object to cameras in section 6.2. Figure 6.9 shows the feasible region of R and d using the values in Table 6.2. If we choose $R = 5200$[mm], R and d fits the feasible region for all d between 29.5 meters and 108 meters. Figure 6.11 shows an example of cell arrangement with $R = 5200$[mm] and camera positions that satisfy the conditions. Based on this arrangement, the view adjustment for each cell can be performed by the same algorithm as the one described in Chapter 5, for example.

Note that the discussion in section 6.2 is based on the zoom value that captures each cell with the exact spatial resolution s. Namely, the algorithm in Chapter 5 only required that each view frustum of a camera contains a common view

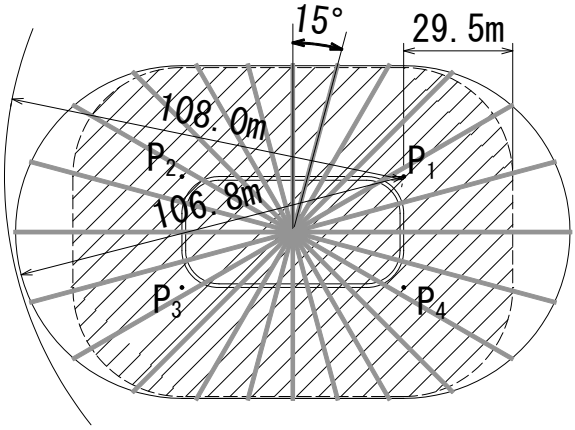


Figure 6.10: Candidate region to mount active cameras for the viewing directions. We will place one camera on each of the thick gray radial lines. Hatched area designates an area where cameras cannot be placed due to the constraint described in section 6.2.2. P_1 through P_4 are corners of a rectangle circumscribing the target area.

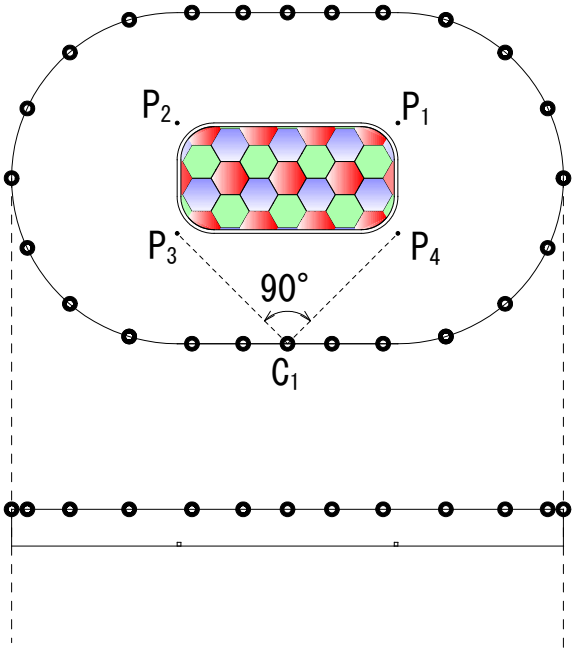


Figure 6.11: An example of the active camera arrangement and the cell arrangement with $R = 5200$[mm]. Small circles represent the camera positions. P_1 through P_4 are corners of a rectangle circumscribing the target area.

volume of each cell, thus it was sometimes possible to choose longer focal length (zoom up further). However, we must use the zoom values by Eq. (6.7) to ensure continuous 3D video capture.

Finally we confirm that the view adjustment step succeeds for this example. As discussed in section 6.2.3, proper zoom values can be found within the mechanical limit of EVI-HD7V. The largest pan motion is required for the camera designated as C_1 in Fig. 6.11. It does not span more than 90 degrees, regardless of which cells the camera views. It is within the limit of EVI-HD7V, which is 200 degrees as shown in Table 6.1.

After the camera view adjustment, the cell-based calibration and tracking can be performed by the same way as described in Chapter 5.

6.4 Summary

A figure skater capture system was designed based on the cell-based framework. The algorithm described in Chapter 5 was used to allow a skater to move freely. We have discussed the dynamic characteristics of the active cameras in more detail to induce the necessary condition of the cell size and active camera locations with given resources and scenario. With proper selection of the cell size, cell-based 3D video capture of a figure skater can be realized using off-the-shelf active cameras.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we proposed a cell-based 3D video capture method. Existing 3D video capture methods use static cameras. Such systems are not scalable to the size of capture space. Especially when an object moves in a wide area, most of the resources — cameras, computers, and storage devices — are consumed to capture nothing but background, producing no effective information of the object. Use of active cameras can use limited resources more effectively, and virtually extends capture space. That approach requires accurate calibration of active cameras and real-time tracking of object satisfying requirements for 3D video generation.

In Chapter 3, we proposed the cell-based framework as a solution to the 3D video capture problem using active cameras. The cell-based framework first divides the space into cells, sets up camera control values for each cell, calibrates them for each cell, then conducts object tracking based on the cells.

Based on the framework, Chapter 4 proposed a 3D video capture algorithm which can be applied when the path of the object movement is given in advance to capture. The path is divided into cells and camera control is performed by a pseudo-optimal assignment of camera modes to each point of the path.

Chapter 5 proposed another algorithm to capture an object that moves freely on a flat floor. The target area was divided into regular hexagons. It was shown that capturing the nearest three cells to the object can ensure continuous object capture with at least one third of the cameras.

Chapter 6 showed an example design of a figure skater capture system based on the algorithm proposed in Chapter 5 in a more realistic situation. It adopted a constant-speed motion model to reflect dynamic characteristics of off-the-shelf

active cameras and induced sufficient conditions for applying the algorithm. It also showed guidelines for active camera positioning reflecting dynamic characteristics of the active cameras.

7.2 Future Work

7.2.1 Multiple Object Capture

Capture of two or more objects causes occlusion between them. The visual coverage introduced in Chapter 3 must be refined to deal with it. On the other hand, when the two objects get away from each other, the system should assign subsets of the active cameras to view two objects in different places. This is potentially a new problem in the camera control process. Additionally, when multiple objects moves independently to each other, it increases the dimension of the object locomotion domain **D**, introduced in Chapter 3. For example, when two objects moves freely on a flat floor, we get four-dimensional space as the object locomotion domain. We will need a new strategy for the cell generation process as well.

7.2.2 Cell-based Viewpoint Planning

Although Chapter 6 showed some guidelines on the static arrangement of the cameras, it was based on a heuristics in existing 3D video studio design methodology, because the target area and the arena was isotropic. When the target area is not isotropic, we predict that non-isotropic camera arrangement reflecting the shape of the target area can improve the visual coverage. According to the cell-based framework, we can evaluate the camera arrangement for each cell statically, by any existing methods for static cameras [SMN+09]. If we were to integrate viewpoint planning process to cell-based algorithms, we would need to define an overall evaluation of the camera arrangement by aggregating such cell-based camera location evaluations, then invent some algorithms to find optimal viewpoints in terms of that criteria.

Additionally, we can consider the use of active cameras which can change their viewpoints. For example, we can consider using crane cameras or cameras on a sliding stage. According to the cell-based framework, sets of viewpoints for each cell can be generated so that Reqs. 2 and 3 are satisfied, as well as their direction and zoom values. Since the translational motions of the cameras require

much larger mechanism than pan/tilt/zoom motions, such motions tends to be slower. It makes the camera control problem even more difficult. It is predicted that more effective camera control strategy is required.

Bibliography

[AAB96] Naoki Asada, Akira Amano, and Masashi Baba. Photometric calibration of zoom lens systems. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 186–190, 1996.

[BB07] A. Bakhtari and B. Benhabib. An active vision system for multitarget surveillance in dynamic environments. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(1):190–198, February 2007.

[BSM+03] Hector M. Briceño, Pedro V. Sander, Leonard McMillan, Steven Gortler, and Hugues Hoppe. Geometry videos: a new representation for 3D animations. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 136–146, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[CJM03] Chi-Wei Chu, Odest Chadwicke Jenkins, and Maja J Mataric. Markerless kinematic model and motion capture from volume sequences. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:475, 2003.

[CRS98] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17(2), pages 167–174, 1998.

[dAST+08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 98:1–98:10, New York, NY, USA, 2008. ACM.

[EDDM+08] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *Computer Graphics Forum*, 27(2):409–418, 2008.

[ES04] C. Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[FP06] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. In *Proceedings of the European Conference on Computer Vision*, pages 564–577, 2006.

[HZ04] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[IB98] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[Int08] International Skating Union. *Special Regulations & technical Rules Single & Pair Skating and Ice Dance*. International Skating Union, 2008. Rule 342 Required rinks.

[IW05] Adrian Ilie and Greg Welch. Ensuring color consistency across multiple cameras. *Computer Vision, IEEE International Conference on*, 2:1268–1275, 2005.

[JKKW06] Ankur Jain, D. Kopell, K. Kakligian, and Yuan-Fang Wang. Using stationary-dynamic camera assemblies for wide-area video surveillance and selective attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 537 – 544, June 2006.

[KRN97] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, 4(1):33–47, 1997.

[KS00] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[KSA+01] I. Kitahara, H. Saito, S. Akimichi, T. Onno, Y. Ohta, and T. Kanade. Large-scale virtualized reality. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR2001), Technical Sketches*, 2001.

[KWM03] Junji Kondou, Xiaojun Wu, and Takashi Matsuyama. Calibration of partially-fixed viewpoint active camera. *IPSJ SIG Notes. CVIM (in Japanese)*, 2003(36)(137-19):149–156, March 2003.

[Lau94] Aldo Laurentini. The visual hull concept for silhouette based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 31–42, New York, NY, USA, 1996. ACM.

[Mat98] Takashi Matsuyama. Cooperative distributed vision – dynamic integration of visual perception, action, and communication –. In *Proceedings of Image Understanding Workshop*, pages 365–384. Monterey CA, 1998. (invited paper).

[MNMM06] Tomoyuki Mukasa, Shohei Nobuhara, Atsuto Maki, and Takashi Matsuyama. Finding articulated body in time-series volume data. In Francisco Perales and Robert Fisher, editors, *Articulated Motion and Deformable Objects*, volume 4069 of *Lecture Notes in Computer Science*, pages 395–404. Springer Berlin / Heidelberg, 2006.

[MTG97] Saied Moezzi, Li-Cheng Tai, and Philippe Gerard. Virtual view generation for 3D digital video. *IEEE MultiMedia*, pages 18–26, 1997.

[MWTN04] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real–time 3D shape reconstruction, dynamic 3D mesh deformation, and high fi-

delity visualization for 3D video. *Computer Vision and Image Understanding*, 96(3):393–434, 2004.

[NM04] Shohei Nobuhara and Takashi Matsuyama. Heterogeneous deformation model for 3D shape and motion recovery from multi-viewpoint images. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 566–573, September 2004.

[O'R87] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[SCD+06] Steven M. Seitz, B. Curless, J. Diebel, D. Scharstein, and Richard Stepen Szeliski. Comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[SD96] Steven M. Seitz and Charles R. Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 21–30, New York, NY, USA, 1996. ACM.

[SH05] J. Starck and A. Hilton. Virtual view synthesis of people from multiple view video sequences. *Graphical Models*, pages 600–620, 2005.

[SH07a] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, pages 21–31, 2007.

[SH07b] Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27:21–31, 2007.

[SMN+09] Jonathan Starck, Atsuto Maki, Shohei Nobuhara, Adrian Hilton, and Takashi Matsuyama. The multiple-camera 3D production studio. *IEEE Transactions on Circuits and System for Video Technology*, 19(6):856–869, June 2009.

[SMP05] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A convenient multicamera self-calibration for virtual environments. *Presence: Teleoperators and Virtual Environments*, 14(4):407–422, 2005.

[SR08] Eric Sommerlade and Ian Reid. Information-theoretic active scene exploration. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[ST03] John R. Spletzer and Camillo J. Taylor. Dynamic sensor planning and control for optimally tracking targets. *Journal of Robotics Research*, January 2003.

[SVZ00] Dan Snow, Paul Viola, and Ramin Zabih. Exact voxel occupancy with graph cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 345–352, 2000.

[TAL+07] Christian Theobalt, Naveed Ahmed, Hendrik Lensch, Marcus Magnor, and Hans-Peter Seidel. Seeing people in different light-joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics*, 13:663–674, 2007.

[TAT95] Konstantinos A. Tarabanis, Peter K. Allen, and Roger Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, 1995.

[THM10] Takeshi Takai, Adrian Hilton, and Takashi Matsuyama. Harmonised texture mapping. In *Proceedings of the Fifth International Symposium on 3D Data Processing, Visualization and Transmission*, May 2010.

[TNM08] Tony Tung, Shohei Nobuhara, and Takashi Matsuyama. Simultaneous super-resolution and 3D video using graph-cuts. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2008.

[TNYM06] Takeshi Takai, Shohei Nobuhara, Hiromasa Yoshimoto, and Takashi Matsuyama. 3D video technologies: Capturing high fidelity full 3D shape, motion, and texture. In *Proceedings of International Workshop on Mixed Reality Technology for Filmmaking (in cooperation with ISMAR 2006)*, 10 2006.

[TTA95] KA Tarabanis, RY Tsai, and PK Allen. The MVP sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation*, 11(1):72–85, 1995.

[TV08] Johan Tangelder and Remco Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39:441–471, 2008. 10.1007/s11042-007-0181-0.

[UM05] N. Ukita and T. Matsuyama. Real-time cooperative multi-target tracking by communicating active visionagents. *Computer Vision and Image Understanding*, 97(2):137–179, 2005.

[WL01] Sebastian Weik and C.-E. Liedtke. Hierarchical 3D pose estimation for articulated human body models from a sequence of volume data. In Reinhard Klette, Shmuel Peleg, and Gerald Sommer, editors, *Robot Vision*, volume 1998 of *Lecture Notes in Computer Science*, pages 27–34. Springer Berlin / Heidelberg, 2001.

[YUK06] Sofiane Yous, Norimichi Ukita, and Masatsugu Kidode. Multiple active camera assignment for high fidelity 3D video. *International Conference on Computer Vision Systems*, page 43, 2006.

[YYM10] Tatsuhisa Yamaguchi, Hiromasa Yoshimoto, and Takashi Matsuyama. Cell-based 3D video capture method with active cameras. In Remi Ronfard and Gabriel Taubin, editors, *Image and Geometry Processing for 3-D Cinematography*. Springer, 2010.

[YYMM09] Hiromasa Yoshimoto, Tatsuhisa Yamaguchi, Atsuto Maki, and Takashi Matsuyama. A cell-based 3D video capturing method with active cameras. *The IEICE Transactions on Information and Systems (Japanese Edition)*, J92-D(9):1579–1590, September 2009.

[YYNM10] Tatsuhisa Yamaguchi, Hiromasa Yoshimoto, Shohei Nobuhara, and Takashi Matsuyama. Cell-based 3D video capture of a freely-moving object using multi-viewpoint active cameras. *IPSJ Transactions on Computer Vision and Applications*, 2:169–184, November 2010.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

List of Publications

Journal Paper

- Hiromasa Yoshimoto, Tatsuhisa Yamaguchi, Atsuto Maki and Takashi Matsuyama. A cell-based 3D video capturing method with active cameras. In *The IEICE Transactions on Information and Systems (Japanese Edition)*, J92-D(9):1579–1590, September 2009.

- Tatsuhisa Yamaguchi, Hiromasa Yoshimoto, Shohei Nobuhara and Takashi Matsuyama. Cell-based 3D video capture of a freely-moving object using multi-viewpoint active cameras. In *IPSJ Transactions on Computer Vision and Applications*, Vol.2:169–184, November 2010.

International Conference

- Tatsuhisa Yamaguchi, Bennett Wilburn and Eyal Ofek. Video-based modeling of dynamic hair. In *Proceedings of the Third Pacific Rim Symposium on Image and Video Technology (PSIVT 2009)*, pages 585–596, Tokyo, Japan, January 2009.

- Tatsuhisa Yamaguchi, Shohei Nobuhara and Takashi Matsuyama. Cell-based object tracking method for 3D shape reconstruction using multi-viewpoint active cameras. In *Proceedings of the Ninth IEEE International Workshop on Visual Surveillance (VS2009)*, Kyoto, Japan, October 2009.

Book Chapter

- Tatsuhisa Yamaguchi, Hiromasa Yoshimoto, and Takashi Matsuyama. Cell-based 3D video capture method with active cameras. In *Image and Geometry Processing for 3-D Cinematography*, pages 171–191, Springer, 2010

.

Domestic Conference

- Tatsuhisa Yamaguchi, Satoshi Nishiguchi, Koh Kakusho, Michihiko Minoh. A method to correct distortion of projective grid spaces for generating images from student's viewpoint with weakly-calibrated cameras. In *Proceedings of the 49th Conference on Systems, Control and Information (SCI'05)*, Kyoto, Japan, May 2005 (in Japanese).